Tallinna Ülikool

Kvalitatiivne digihumanitaaria

Jaagup Kippar

Tallinn 2019

Sisukord

Sissejuhatus

Kvalitatiivse uurimisviisi puhul saab andmeid mitmel moel esitades ja võrreldes ka siis kasutatava tulemuseni jõuda, kui meil ühele statistilisele meetodile toetudes pole võimalik piisavat usaldusväärset järeldust saavutada. Digivaldkonna areng pakub märgatavalt võimalusi andmete illustreerimiseks, seoste leidmiseks ja esitamiseks ning sealtkaudu mõistmiseks, et mil moel miski protsess toimib ning millised variatsioonid sel on. Siinses materjalis näidatakse, kuidas ümber käia andmepuude ja -võrgustikega, kuidas nende abil välja tuua seoseid, mis tabeli või loeteluna andmetele peale vaadates ehk kohe välja ei paista. Õpitakse andmete esitamiseks animatsioone looma, kus aja- või mõnel muul teljel väärtusi vahetades näeb protsesside ja nendega kaasnevate nähtuste suunda. Paljude andmetega käib kaasas asukoht - nii tutvutakse kaardirakenduse võimalustega. Muusikat saab nii uurida kui illustreerimisel kasutada. Märgatav osa näiteid toetub Eesti Kirjandusmuuseumi Eesti Rahvaluule Arhiivist kasutamiseks antud eesti regiviiside andmebaasi koopiale ning autor tänab Arhiivi näidete mitmekesistamise võimaluse eest.

Andmepuud

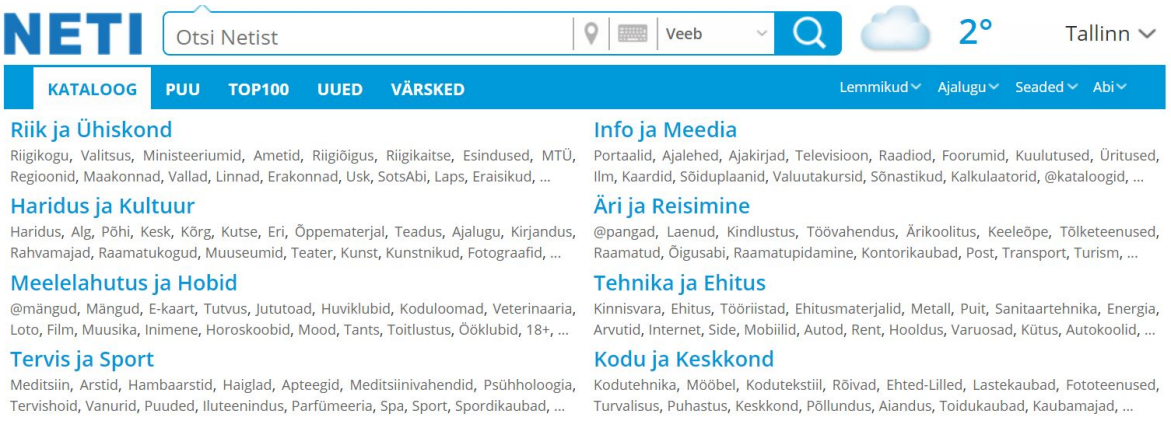
Uurijad püüavad leitut süstematiseerida - ikka selleks, et parem ülevaade oleks ning kättesaadavate vahendite abil õnnestuks muidu segaseks jääma kippuvat andmestikku hallata.

Vajaliku leidmine

Andmepuu on hea moodus suure hulga teabe sees orienteerumiseks. Tüüpiliseks näiteks on veebilehestik. Avalehele saab kirjutada tutvustuse ja tähtsamad andmed. Vähegi keerukamas lehestikus aga kasutaja paratamatult ei vaja kõiki andmeid kohe kätte. Menüüvalikuga kannatab lisada mõned alateemad, selgema süsteemiga menüü puhul ka mõnikümmend teemaviidet avalehele nii, et sealt suudab sobivalt edasi liikuda.

Disainerid rõhutavad vahel "kolme kliki reeglit" - ehk siis kasutajale tundub veebileht mugav, kui ta saab omale vajaliku lehe kätte avalehelt kolme klõpsuga. Sealtkaudu veidi arvutades leiab, et erisuguse hargnemisteguri ehk lehtedel olevate viidete arvu puhul võib avalehelt kolme klõpsuga jõutavate lehtede arv päris erinev olla. Kui pealehel vaid ligikaudu viis suuremat valikut nagu väiksematel tutvustuslehtedel tavaks, siis pääseb esimese vajutusega viiele, teisega 5x5 ehk 25le ning kolmandaga 125 lehe peale. Aruka läbi mõtlemise peale õnnestub päris palju kasutajale vajalikku teavet veidi rohkem kui sajale lehele paigutada.

Kus andmekogused suuremad, seal tuleb nende kättesaadavamaks muutmiseks ka igale lehele rohkem valikuid anda. Otsimislehe neti.ee avalehel on vähemasti viiskümmend valikut ning alanejate juures ligikaudu sama palju. Nii pääseb kolme klõpsuga juba 50x50x50 ehk 125000 lehe peale, mis on märgatavalt suurem kogus.



Kui tõsiselt tarviliku sisu jaoks ollakse valmis veel teised kolm klõpsu ka tegema, siis ühtlaselt jaotatud ja puusse paigutatud andmestiku korral on võimalik orienteeruda andmetes, kus veebilehti on rohkem kui inimesi maakeral.

Harjutus

- Otsi mõni puukujuline andmestik, vaata, mida on sealt võimalik välja lugeda
- Jälgi, mitmeks harunetakse iga taseme juures, mis on vähim ja suurim hargnemistegur
- Koosta omapoolne puukujuline andmestik

Järgnevustest moodustatud puu

Üheks mooduseks andmetega lähemaks tutvumiseks on sarnaste osade või lõikude otsimine olgu kirjalike tekstide, meloodiate või arvuliste mõõtmistulemuste jada juures. Kindla pikkusega lõike on hea võrrelda, loendada ja järjestada. Sellisel puhul loetakse täiesti erinevaks aga ka juba ühe elemendi võrra erinevat jada. Andmepuu võimaldab hakkama saada erisuguste lõigupikkustega ning nendegi juures levinumad kohad välja tuua ja ettepoole järjestada.

Tutvustuseks mõned näited. Jadaks on tähed sõnas - hiljem saab aga sarnaselt järjestada sõnaliike, lauseliikmeid, noote või tööoperatsioone. Esitatavateks sõnadeks siin “tere” ja “tervis”. Ehk siis esimesed kolm tähte samad, edasi tuleb hargnemine.

```
tere
tervis
```

Taande kaudu puusse tõstetuna näevad sõnad välja nõnda

```
t
 e
  r
   e
   v
    i
     s
```

Juurde kannatab kirjutada ka iga koha sageduse

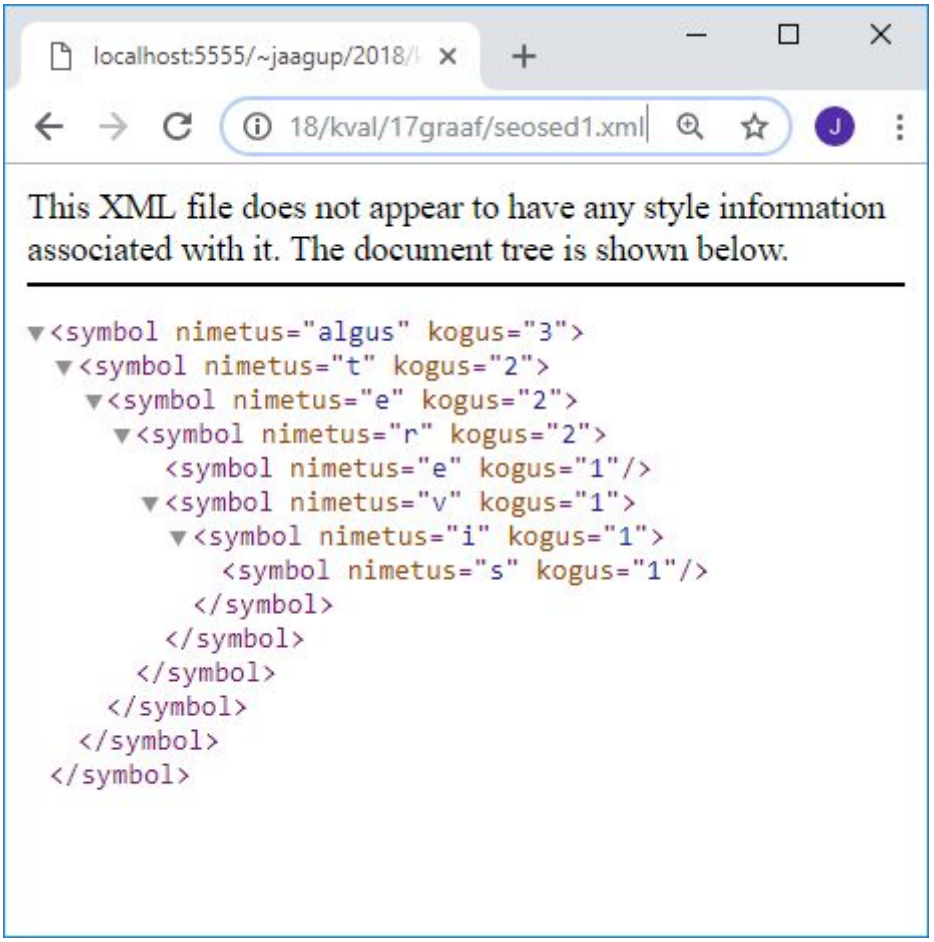
```
t 2
 e 2
  r 2
   e 1
   v 1
    i 1
     s 1
```

XML

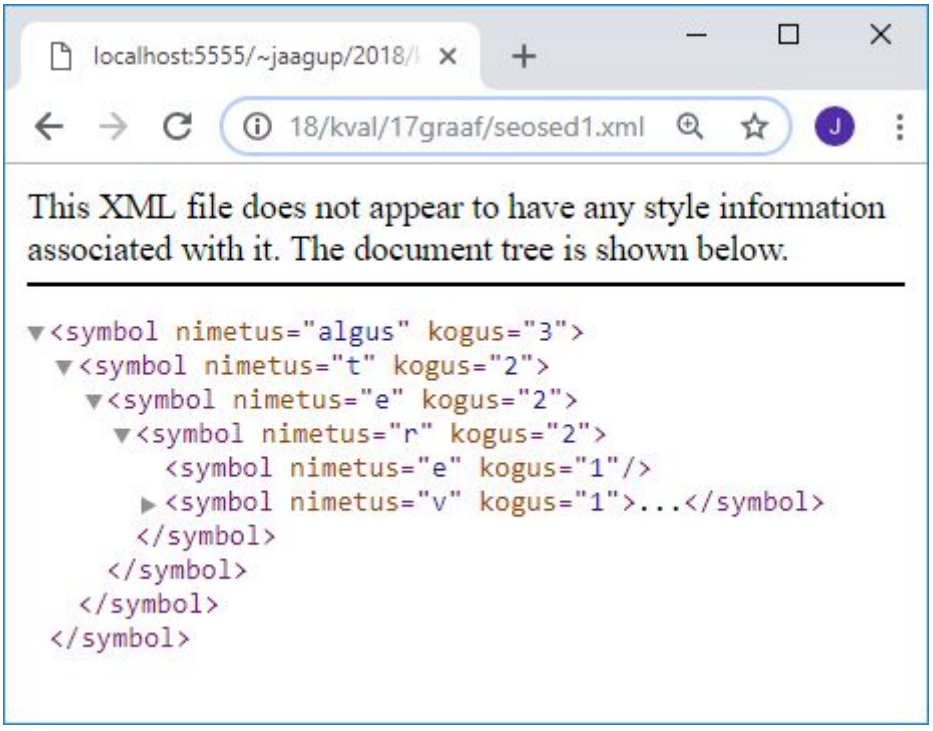
Andmete mugavamaks lugemiseks silmade ja masinaga loodi 2000. aastaks XML keel

```
<?xml version='1.0' ?>
<symbol nimetus="algus" kogus="2">
   <symbol nimetus="t" kogus="2">
      <symbol nimetus="e" kogus="2" >
         <symbol nimetus="r" kogus="2">
            <symbol nimetus="e" kogus="1" />
            <symbol nimetus="v" kogus="1">
               <symbol nimetus="i" kogus="1" >
                  <symbol nimetus="s" kogus="1" />
               </symbol>
            </symbol>
         </symbol>
       </symbol>
   </symbol>
</symbol>
```

Näeb välja eelmisest kirjum, kuid nõnda on igasugu lisaandmete tarbeks kohad olemas. Ühtlasi võimaldab veebilehitseja XML puu harusid kokku ja lahku klõpsida vastavalt vajadusele.



Allolevas näites alates v-st hargnev alampuu on kinni.



Harjutus

- Paiguta tekstina puu kujuliselt sõnad "tere", "tervis" ja "tervitus"
- Kopeeri sõnade "tere" ja "tervis" **tähtede** XML-esitus omaette faili. Veendu, et saad tagumist sõnapoolt brauseris avada ja sulgeda. Faili laiendiks .xml
- Lisa puusse sõna "tervitus"
- Koosta uus tühi puu. Pane näitele sarnaselt XML-andmepuusse sõnade te-re, ter-vis ning ter-vi-tus **silbid**

Sõnade tere, tervis ja tervitus tähed

```
<?xml version='1.0' ?>
<symbol nimetus="algus" kogus="3">
   <symbol nimetus="t" kogus="3">
      <symbol nimetus="e" kogus="3">
          <symbol nimetus="r" kogus="3">
            <symbol nimetus="e" kogus="1" />
            <symbol nimetus="v" kogus="2">
                <symbol nimetus="i" kogus="2" >
                  <symbol nimetus="s" kogus="1" />
                     <symbol nimetus="t" kogus="1">
                      <symbol nimetus="u" kogus="1">
                         <symbol nimetus="s" kogus="1" />
                      </symbol>
                     </symbol>
                </symbol>
```

```
            </symbol>
          </symbol>
        </symbol>
    </symbol>
</symbol>
```

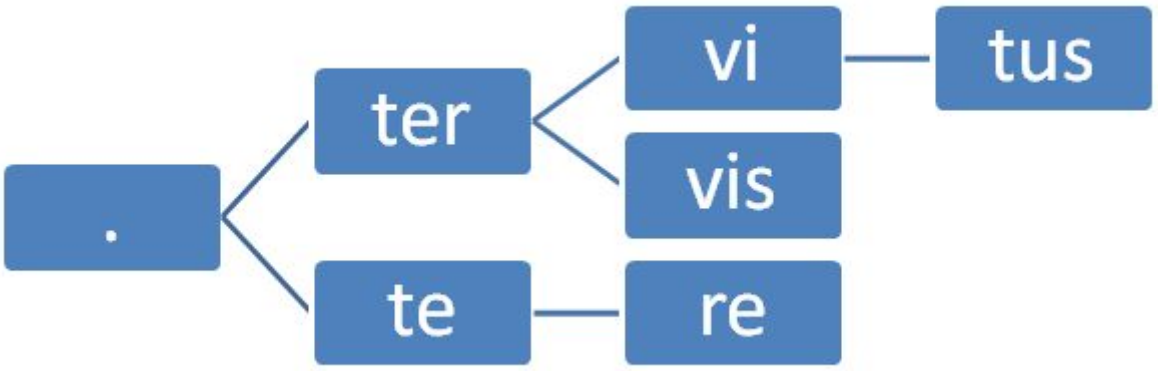
Sõnade tere, tervis ja tervitus silbid

```
<?xml version='1.0' ?>
<silp nimetus="algus" kogus="3">
  <silp nimetus="te" kogus="1">
    <silp nimetus="re" kogus="1">
       </silp>
  </silp>
  <silp nimetus="ter" kogus="2">
    <silp nimetus="vi" kogus="1">
         <silp nimetus="tus" kogus="1">
         </silp>
       </silp>
       <silp nimetus="vis" kogus="1">
       </silp>
  </silp>
</silp>
```

Puu Microsoft Wordi Smart Art vahenditega. Ülaltasemeks on alustamiseks punkt, taanded tehtud tabulaatoritega. Smart Art kuvab puu joonisena
Sisend:

```
.
        ter
                vi
                        tus
                vis
        te
                re
```

Tulemuseks tehtud joonis:



JSON

XMList veidi lühem on JSON (JavaScript Object Notation), kus samuti võimalik masinale arusaadavalt andmete struktuur üles märkida ning pärast sobivad väärtused välja küsida. Sama kahe sõna näide, iga tähe alamelemendiks (tähele vastavaks objektiks) on vastava tähe esinemise kogus (praegu lõpus) ning loetelu tema sees olevatest tähtedest. Sellisena suudab andmed sisse võtta näiteks Pythoni keel ning siis all näites küsitakse, et mitu korda vastavat järgnevust (praegu “te”) esines.

```
andmed={
  't':{
    'e':{
      'r':{
        'e':{
         'kogus':1
        },
        'v':{
           'i':{
             's':{
               't':{
                 'kogus':1
               },'kogus':1
             },'kogus':1
           },'kogus':1
        },'kogus':2
      },'kogus':2
    },'kogus':2
  },'kogus':2
}
print(andmed["t"]["e"]["kogus"])

jaagup@praktika1 ~/public_html/2018/kval/18puu $ python3.5 puu2.py
2
```

Sulge lugedes ja süvenedes paistab, et “te” kogus on kõrvuti elemendiga ‘r’ - samuti, nagu “ter”-i kogus on kõrvuti elementidega ‘e’ ja ‘v’. Ehk siis praeguse üleskirjutuskuju puhul peame arvestama, et "kogus" on erilise nimega väli, mis ütleb, mitu korda vastavat tähte oli, mille elemendi sees kogus kirjas on.

Harjutus

- Pane eelmisega sarnaselt JSON-puusse sõnade te-re, ter-vis ja ter-vi-tus silbid.
- Kuva silbi "ter" esinemiskordade arv
- Kuva silbijärjendi "ter"-"vi" esinemiskordade arv

Puu loomine programmiga

Mõne või suurema püsivuse korral ka mõnesaja järgnevuse andmete puusse sättimine tuleb käsitsi välja. Suurematest andmestikest aga tuhandete või kümnete tuhandete sarnasuste leidmiseks on kasulik tarkvara luua või kasutada. Kuna põhioperatsioonid saab mõne(teistkümne) käsu abil programmeerimiskeele abil ette anda, siis siin seda teed pidi läheme, koostame puud loovat programmi lihtsamast keerulisema poole. Hiljem saab vastavalt andmete tüübile vajalikke täiendusi teha.

Algus võimalikult lihtne ja lühike. Muutujasse nimega `andmed` hakkame koguma sõnade tähtede andmepuud. Muutuja `plokk` tegeleb selle kohaga, kus parajasti andmepuus ollakse. Et alustame puu juurest, siis algul plokk=andmed. Puusse paigutatavaks täheks on kõigepealt sõna esimene täht (Pythonis järjekorranumbriga 0). Tuleviku huvides siia tingimus, et kui täht on juba olemas, siis saab selle tähe koguse loendurit suurendada. Esialgu aga selle asemel käsklus pass ehk käsk puudub. Kui tähte plokis ei ole (nagu praegu alguses), siis minnakse else-ossa, luuakse ploki sisse võti parasjagu paigutatava tähe (praegu ‘h’) väärtusega ning määratakse selle tähe koguseks 1.

```
andmed={}
sona="hei"
taht=sona[0]
plokk=andmed
if taht in plokk:
  pass
else:
  plokk[taht]={'kogus':1}

print(andmed)
```

Väljatrüki tulemus:

```
jaagup@praktika1 ~/public_html/2018/kval/18puu $ python3.5 puu3.py
{'h': {'kogus': 1}}
```

Täiendusena suurendatakse tähe olemasolul plokis vastava tähe kogust. Nt. kui puusse paigutatakse kaks h-tähega algavat sõna, siis saab puu juurest väljuva h-tähe haru koguseks 2. Pärast tähe paigutamist puus suunatakse muutuja plokk näitama selle tähe kohale

```
 plokk=plokk[taht]
```

Nii on võimalik väärtused koos kogustega puusse kirjutada. Väljatrükiks tuleb küsida võtmed sobivas järjekorras ning lõppu kogus vastavate korduste arvu teada saamiseks

```
print(andmed["h"]["e"]["kogus"])
```

Kood tervikuna

```
andmed={}
sona="hei"
plokk=andmed
for taht in sona:
 if taht in plokk:
  plokk[taht][kogus]+=1
 else:
  plokk[taht]={'kogus':1}
 plokk=plokk[taht]

print(andmed)
print(andmed["h"]["e"]["kogus"])
```

ja väljund. Sulge lugedes näha, et h sees on e ning e sees i, juures kõigil kogus väärtusega 1

```
jaagup@praktika1 ~/public_html/2018/kval/18puu $ python3.5 puu4.py
{'h': {'kogus': 1, 'e': {'kogus': 1, 'i': {'kogus': 1}}}}
1
```

Sama koodilõik näitega, kus puusse paigutatakse sõnad “hei” ja “hi”. Järjestuse “h” koguseks tuleb siis 2, “he” puhul aga 1.

```
andmed={}
sonad=["hei", "hi"]
for sona in sonad:
 plokk=andmed
 for taht in sona:
  if taht in plokk:
   plokk[taht]["kogus"]+=1
  else:
   plokk[taht]={'kogus':1}
  plokk=plokk[taht]

print(andmed)
print(andmed["h"]["kogus"])
print(andmed["h"]["e"]["kogus"])
```

```
jaagup@praktika1 ~/public_html/2018/kval/18puu $ python3.5 puu5.py
{'h': {'i': {'kogus': 1}, 'e': {'i': {'kogus': 1}, 'kogus': 1}, 'kogus': 2}}
2
1
```

Harjutus

- Pane näited käima
- Vaata ühe sõnaga näite tulemust mõne muu sõna puhul
- Pane teise näitesse mitu omapoolset sõna sisse ja jälgi tulemust.

Sõnaliikide puu

Eelnev näide tehti lihtsuse mõttes sõnade tähtede peal. Sarnased puusse paigutamised tulevad aga ette igal pool, kus järgnevustega tegemist. Järgnevused aga digihumanitaarias tähtis mõõdetav/võrreldav nähtus - olgu tegemist sündmustüüpidega filmis või ajaloos, nootide/intervallidega muusikapalas või keele uuringu juures ette tulevate üksustega. Puu harude sageduste leidmine võimaldab meil ennustada jada järgmist elementi või siis lihtsalt iseloomustada ja võrrelda nähtusi.

Alustuseks paar lauset, milles leiduvate sõnade liigid välja trükitakse.

Klass `Text` paketist `estnltk` võimaldab küsida sõnade liigid lausete kaupa.

```
from estnltk import Text
laused=["Juku tuli kooli", "Juku tuli joostes kooli", "jookseb linna"]
for lause in laused:
  print(Text(lause).postags)

jaagup@praktika1 ~/public_html/2018/kval/17graaf $ python3.5 sonaliigid1.py
['H', 'V', 'S']
['H', 'V', 'V', 'S']
['V', 'S']
```

Sõnaliikide seletused:

H - pärisnimi
V - tegusõna (verb)
S - nimisõna (substantiiv)

ehk siis lause “Juku tuli kooli” puhul on “Juku” H ehk pärisnimi, “tuli” tegusõna ning “kooli” nimisõna.

Nende põhjal loodud puu näeb välja järgmine:

```
H
 V
  S
  V
   S
V
 S
```

Programmilõik töötab sarnaselt eelmisele, ainult et endise tähtederea asemel on nüüd sõnaliikide rida ning puu teeks juurest leheni on endise sõna asemel terve lause.

```
from estnltk import Text
laused=["Juku tuli kooli", "Juku tuli joostes kooli"]
andmed={}
for lause in laused:
```

```
  plokk=andmed
  for sonaliik in Text(lause).postags:
    if sonaliik in plokk:
      plokk[sonaliik]['kogus']+=1
    else:
      plokk[sonaliik]={'kogus':1}
    plokk=plokk[sonaliik]

print(andmed)

jaagup@praktika1 ~/public_html/2018/kval/18puu $ python3.5 puu6.py
{'H': {'V': {'V': {'kogus': 1, 'S': {'kogus': 1}}, 'kogus': 2, 'S': {'kogus': 1}}, 'kogus':
2}}
```

Sama vastus loetavuse huvides trepituna

```
{'H':
  {'kogus': 2,
   'V': {'kogus': 2,
         'S': {'kogus': 1},
         'V':
          {'kogus': 1,
           'S': {'kogus': 1}}
   }
 },
 'V':
  {'kogus': 1,
  'S': {'kogus': 1}
  }
}
```

Harjutus

- Pane näide käima
- Katseta tööd ja tulemust omapoolse paari veidi sarnase lausega

Väljundi kohendamine

Mälus oleva andmepuu saab print-käsu abil eelnevalt näha oleva JSONi kujul välja trükkida. Eriti aga suuremate andmete puhul tasub parema ülevaate saamiseks sealt sobivaid väljavõtteid teha. Siin näites luuakse eraldi alamprogramm nimega `tryki`, kus kuvatakse puus etteantud koha (sõlme) alamsõlmed koos nende kogustega. Allpool kuvatakse vasted juurelemendile (kahel korral pärisnimi, ühel korral verb, järjekord pole esialgu tähtis), siis pärisnime alamelement (verb kahel korral) ning lõpuks järgnevuse HV alamelemendid (ühel korral S ja ühel korral V)

```
from estnltk import Text
laused=["Juku tuli kooli", "Juku tuli joostes kooli", "jookseb linna"]
andmed={}
for lause in laused:
```

```
  plokk=andmed
  for sonaliik in Text(lause).postags:
    if sonaliik in plokk:
      plokk[sonaliik]['kogus']+=1
    else:
      plokk[sonaliik]={'kogus':1}
    plokk=plokk[sonaliik]

print(andmed)

def tryki(solm):
  for voti in solm:
    if not voti=="kogus":
      print(voti+"("+str(solm[voti]["kogus"])+")")

tryki(andmed)
print()

tryki(andmed["H"])
print()
tryki(andmed["H"]["V"])
```

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus1.py
{'V': {'S': {'kogus': 1}, 'kogus': 1}, 'H': {'kogus': 2, 'V': {'S': {'kogus': 1}, 'kogus':
2, 'V': {'S': {'kogus': 1}, 'kogus': 1}}}}
V(1)
H(2)

V(2)

S(1)
V(1)
```

Viimatises näites näeb korraga vaid ühe taseme elemente ning nad kõik on vastu rea vasakut serva. Puus või alampuus alamelementide jälgimiseks on nad mugav servast sobivale kaugusele treppida - nii pääseb ülalt alla vaadates jälgima, et mis samale tasemele kuulub. Funktsiooni `tryki` viimase käsuna kutsutakse sama funktsioon uuesti välja - ainult et ühe taseme võrra sügavamalt - muidugi ainult juhul, kui seal veel andmeid on.

```
from estnltk import Text
laused=["Juku tuli kooli", "Juku tuli joostes kooli", "jookseb linna"]
andmed={}
for lause in laused:
  plokk=andmed
  for sonaliik in Text(lause).postags:
    if sonaliik in plokk:
      plokk[sonaliik]['kogus']+=1
    else:
      plokk[sonaliik]={'kogus':1}
    plokk=plokk[sonaliik]

print(andmed)

def tryki(solm, tase):
  for voti in solm:
    if not voti=="kogus":
      print((tase*" ")+voti+"("+str(solm[voti]["kogus"])+")")
```

```
      tryki(solm[voti], tase+1)

tryki(andmed, 0)
```

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus2.py
{'H': {'kogus': 2, 'V': {'kogus': 2, 'S': {'kogus': 1}, 'V': {'kogus': 1, 'S': {'kogus':
1}}}}, 'V': {'kogus': 1, 'S': {'kogus': 1}}}
H(2)
 V(2)
  S(1)
  V(1)
   S(1)
V(1)
 S(1)
```

Harjutus

- Pane näited käima. Lisa paar lauset ja vaata puud.
- Loe sisend tekstifailist. Koosta sõna tähtede puu ja trüki see trepituna välja.

XML-väljund

Eelnevalt tutvustatud XML-kuju on mugav vorming andmete jaoks, mida peavad lugema nii masinad kui ka inimesed. Alustuseks väike näide andmete sisse lugemise ja välja kirjutamise kohta. Pythonis XMLiga toimetamise juures on mugav kasutada paketti nimega xml.dom.minidom. Sealne käsklus parseString teeb etteantud ja korrektsel kujul XML-tekstist XML-andmete objekti. Hiljem tagasi teksti kujule saab selle käsuga toxml().

```
from xml.dom.minidom import parseString
p=parseString("<s kogus='2'/>")
print(p.toxml())
```

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus3.py
<?xml version="1.0" ?><s kogus="2"/>
```

Nagu näha, pandi XML-failile kohustuslik versiooninumbriga päis ka juurde.

Järgmine veidi pikem näide tühja dokumendi loomise ning sinna külge elementide paigutamise kohta.

Luuakse tühi dokument

```
d=xml.dom.minidom.Document()
```

tehakse alamelement nimega s

```
p=d.createElement("s")
```

pannakse talle atribuudid nimedega kogus ja sisu väärtustega 2 ja algus

```
p.setAttribute("kogus", "2")
p.setAttribute("sisu", "algus")
```

seatakse eraldi loodud s-element puu juureks.

```
d.appendChild(p)
```

luuakse teine sarnane element

```
p2=d.createElement("s")
p2.setAttribute("sisu", "V")
p2.setAttribute("kogus", "1")
```

pannakse see esimese külge

```
p.appendChild(p2)
```

kuvatakse kogu dokument

```
print(d.toxml())
```

Kood tervikuna

```
import xml.dom.minidom

d=xml.dom.minidom.Document()
p=d.createElement("s")
p.setAttribute("kogus", "2")
p.setAttribute("sisu", "algus")
d.appendChild(p)

p2=d.createElement("s")
p2.setAttribute("sisu", "V")
p2.setAttribute("kogus", "1")
p.appendChild(p2)
print(d.toxml())
```

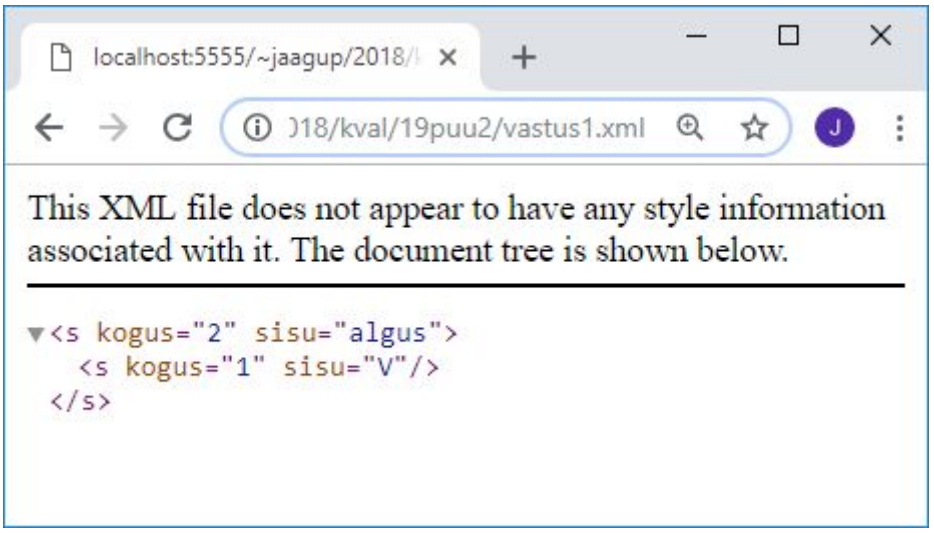
Väljund

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus4.py
<?xml version="1.0" ?><s kogus="2" sisu="algus"><s kogus="1" sisu="V"/></s>
```

Tekstifaili saadetud väljund

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus4.py > vastus1.xml
```

Selle vaatamine brauseris



Nüüd näitena kaks eelmist võimalust kokku. Ehk siis kõigepealt koostatakse lause sõnaliikidest puu ning edasi kantakse see üle XML-kujule. Funktsioon kannaYle saab üheks parameetriks koha algses mälupuus ning teiseks andmepuu sama koha XMLi puus. Esialgu tõstab üle ainult selle koha otsesed alamelemendid ning kommentaaridena on alles veel varasemad väljatrükikäsklused

```
from estnltk import Text
import xml.dom.minidom
laused=["Juku tuli kooli", "Juku tuli joostes kooli", "jookseb linna"]
andmed={}
for lause in laused:
  plokk=andmed
  for sonaliik in Text(lause).postags:
    if sonaliik in plokk:
      plokk[sonaliik]['kogus']+=1
    else:
      plokk[sonaliik]={'kogus':1}
    plokk=plokk[sonaliik]

#print(andmed)

d=xml.dom.minidom.Document()
p=d.createElement("s")
p.setAttribute("sisu", "algus")
d.appendChild(p)

def kannaYle(solm, xmlSolm):
  for voti in solm:
    if not voti=="kogus":
      #print((tase*" ")+voti+"("+str(solm[voti]["kogus"])+")")
      p2=d.createElement("s")
      p2.setAttribute("sisu", voti)
      p2.setAttribute("kogus", str(solm[voti]["kogus"]))
      xmlSolm.appendChild(p2)
#       tryki(solm[voti], tase+1)

kannaYle(andmed, p)
print(d.toxml())
```

Funktsiooni väljakutsel anti ette andmepuu ning tühja XML-i puu juurelement. Esimesest kopeeriti teise sealsed alamelemendid koos kogustega

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus5.py
<?xml version="1.0" ?><s sisu="algus"><s kogus="1" sisu="V"/><s kogus="2" sisu="H"/></s>
```

Koodilõigu täiendatud versioon. Funktsioon kannaYle kannab üle etteantud kohast alamelemendid XML-puusse süvitsi, kuni puu lõpuni. Selleks kutsub funktsioon välja iseenese, andes tsükli sees ükshaaval parameetriteks koha alamelementide kohad.

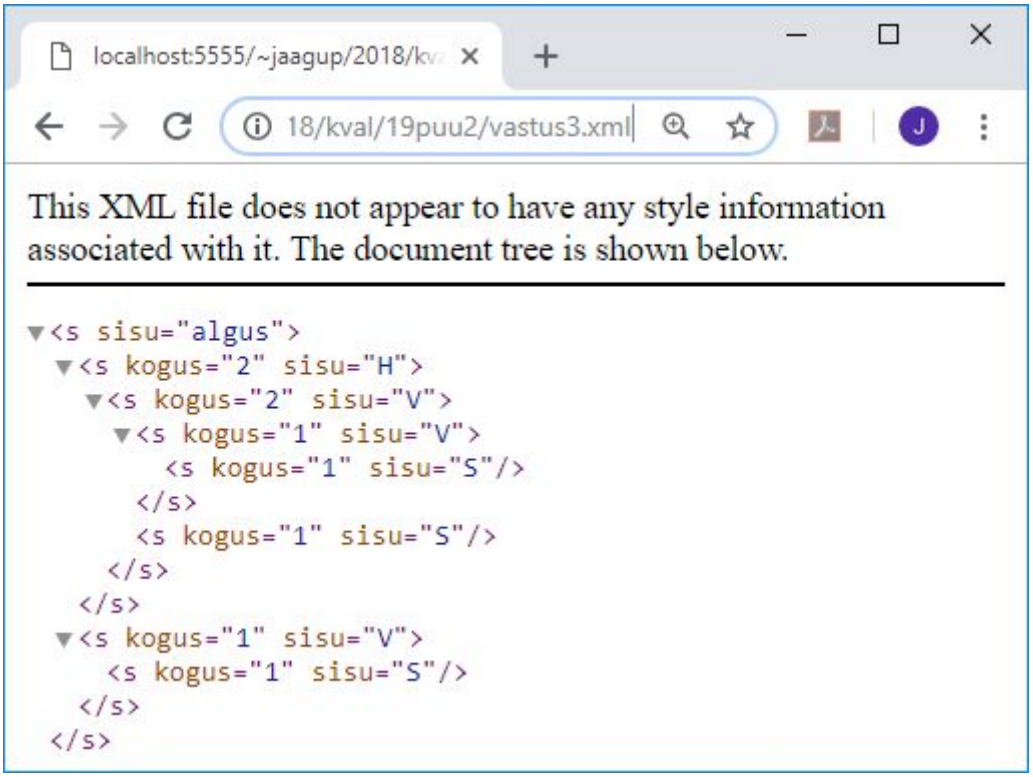
```
from estnltk import Text
import xml.dom.minidom
laused=["Juku tuli kooli", "Juku tuli joostes kooli", "jookseb linna"]
andmed={}
for lause in laused:
  plokk=andmed
  for sonaliik in Text(lause).postags:
    if sonaliik in plokk:
      plokk[sonaliik]['kogus']+=1
    else:
      plokk[sonaliik]={'kogus':1}
    plokk=plokk[sonaliik]


d=xml.dom.minidom.Document()
p=d.createElement("s")
p.setAttribute("sisu", "algus")
d.appendChild(p)

def kannaYle(solm, xmlSolm):
  for voti in solm:
    if not voti=="kogus":
      p2=d.createElement("s")
      p2.setAttribute("sisu", voti)
      p2.setAttribute("kogus", str(solm[voti]["kogus"]))
      xmlSolm.appendChild(p2)
      kannaYle(solm[voti], p2)

kannaYle(andmed, p)
print(d.toxml())
```

Väljund:

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus6.py
<?xml version="1.0" ?><s sisu="algus"><s kogus="1" sisu="V"><s kogus="1" sisu="S"/></s><s
kogus="2" sisu="H"><s kogus="2" sisu="V"><s kogus="1" sisu="V"><s kogus="1"
sisu="S"/></s><s kogus="1" sisu="S"/></s></s></s>
```

ja tulemus trepituna

```
jaagup@praktika1 ~/public_html/2018/kval/19puu2 $ python3.5 valjastus6.py > vastus3.xml
```



Harjutus

- Pane näited käima
- Koosta sarnane XML-puu sõnade tähtedest

Lauseliikmete puu

Puid saab tehnilise poole pealt ehitada igasugu järgnevate andmete põhjal. Tasub lihtsalt vaadata, et mis võiksid ülesande puhul kasulikuks osutuda.

Lauseid uurides jälgitakse mõnikord, et mis rollis mõni sõna on lauses - näiteks nimisõna võib olla nii alus, sihitis kui määrus. Keele automaatne parser annab sõnade kohta hulga teavet - märgendite kirjeldused leiab

https://korpused.keeleressursid.ee/syntaks/dokumendid/syntaksiliides_ee.pdf

Praeguse näite jaoks tähtsad on

@SUBJ subjekt / alus
@FMV finiitne põhiverb / öeldis (pöördeline vorm)
@OBJ objekt / sihitis
@ADVL adverbiaal / määrus

Kõigepealt näide, et kuidas andmed lausete kohta kätte saada. Parser tuleb keskkonda eraldi installida. Kui kõik läheb õnneks, siis on võimalik VISLCG3Parser import-käsklusega sisse lugeda.

```
from estnltk.syntax.parsers import VISLCG3Parser
from estnltk import Text

sisend="Juku tuli kooli. Juku tuli joostes kooli. Jookseb linna"
parser=VISLCG3Parser()
vahetulemus=parser.parse_text(Text(sisend), return_type='vislcg3')
print(vahetulemus)
print("\n".join(vahetulemus))
```

Parserist tulnud vahetulemus kuvatakse praegu nii massiivina kui reavahetustega eraldatud tekstina.

```
jaagup@praktika1 ~/public_html/2018/kval/20puujupid $ python3.5 tekstiandmed1.py
['"<s>"', '', '"<Juku>"', '\t"Juku" L0 S prop sg nom @SUBJ #1->2', '"<tuli>"', '\t"tule" Li
V main indic impf ps3 sg ps af @FMV #2->0', '"<kooli>"', '\t"kool" L0 S com sg adit @ADVL
#3->2', '"<.>"', '\t"." Z Fst #4->4', '"</s>"', '', '"<s>"', '', '"<Juku>"', '\t"Juku" L0 S
prop sg nom @SUBJ #1->2', '"<tuli>"', '\t"tule" Li V main indic impf ps3 sg ps af @FMV
#2->0', '"<joostes>"', '\t"joostes" L0 D @ADVL #3->2', '"<kooli>"', '\t"kool" L0 S com sg
adit @ADVL #4->2', '"<.>"', '\t"." Z Fst #5->5', '"</s>"', '', '"<s>"', '', '"<Jookseb>"',
'\t"jooks" Lb V main indic pres ps3 sg ps af @FMV #1->0', '"<linna>"', '\t"linn" L0 S com
sg adit @ADVL #2->1', '"</s>"', '', '']
"<s>"

"<Juku>"
        "Juku" L0 S prop sg nom @SUBJ #1->2
"<tuli>"
        "tule" Li V main indic impf ps3 sg ps af @FMV #2->0
"<kooli>"
        "kool" L0 S com sg adit @ADVL #3->2
"<.>"
        "." Z Fst #4->4
"</s>"

"<s>"

"<Juku>"
        "Juku" L0 S prop sg nom @SUBJ #1->2
"<tuli>"
        "tule" Li V main indic impf ps3 sg ps af @FMV #2->0
"<joostes>"
        "joostes" L0 D @ADVL #3->2
"<kooli>"
        "kool" L0 S com sg adit @ADVL #4->2
"<.>"
        "." Z Fst #5->5
"</s>"

"<s>"

"<Jookseb>"
        "jooks" Lb V main indic pres ps3 sg ps af @FMV #1->0
"<linna>"
        "linn" L0 S com sg adit @ADVL #2->1
"</s>"
```

Regulaaravaldise abil otsime igale sõna kohta välja @-märgiga algava tunnuse ehk vaste lauseliikmena.

```
from estnltk.syntax.parsers import VISLCG3Parser
from estnltk import Text
import re

sisend="Juku tuli kooli. Juku tuli joostes kooli. Jookseb linna"
parser=VISLCG3Parser()
vahetulemus=parser.parse_text(Text(sisend), return_type='vislcg3')
for rida in vahetulemus:
  if rida=='"<s>"': print("lause algus")
  m=re.findall("@[A-Z]+", rida)
  if m: print(m[0])
```

Tulemus:

```
jaagup@praktika1 ~/public_html/2018/kval/20puujupid $ python3.5 tekstiandmed2.py
lause algus
@SUBJ
@FMV
@ADVL
lause algus
@SUBJ
@FMV
@ADVL
@ADVL
lause algus
@FMV
@ADVL
```

Nüüd juba mitu toimetust üheskoos. Iga lause kohta koostatakse lauseliikmete jada. Jada põhjal luuakse mällu andmepuu ning siis teisendatakse see XML-i kujule ja kuvatakse

```
from estnltk.syntax.parsers import VISLCG3Parser
from estnltk import Text
import xml.dom.minidom

import re

sisend="Juku tuli kooli. Juku tuli joostes kooli. Jookseb linna"
parser=VISLCG3Parser()
vahetulemus=parser.parse_text(Text(sisend), return_type='vislcg3')
laused=[]
for rida in vahetulemus:
  if rida=='"<s>"': margendid=[]
  m=re.findall("@[A-Z]+", rida)
  if m: margendid.append(m[0])
  if rida=='"</s>"': laused.append(margendid)

#print(laused)

andmed={}
for margendid in laused:
  plokk=andmed
  for sonaliik in margendid:
    if sonaliik in plokk:
      plokk[sonaliik]['kogus']+=1
    else:
      plokk[sonaliik]={'kogus':1}
    plokk=plokk[sonaliik]
```

```
#print(andmed)

d=xml.dom.minidom.Document()
p=d.createElement("s")
p.setAttribute("sisu", "algus")
d.appendChild(p)

def kannaYle(solm, xmlSolm):
  for voti in solm:
    if not voti=="kogus":
      p2=d.createElement("s")
      p2.setAttribute("sisu", voti)
      p2.setAttribute("kogus", str(solm[voti]["kogus"]))
      xmlSolm.appendChild(p2)
      kannaYle(solm[voti], p2)

kannaYle(andmed, p)
print(d.toxml())
```

Valminud puu:

```
<?xml version="1.0" ?>
<s sisu="algus">
  <s kogus="2" sisu="@SUBJ">
    <s kogus="2" sisu="@FMV">
      <s kogus="2" sisu="@ADVL">
        <s kogus="1" sisu="@ADVL"/>
      </s>
    </s>
  </s>
  <s kogus="1" sisu="@FMV">
    <s kogus="1" sisu="@ADVL"/>
  </s>
</s>
```

Harjutus

- Pane näide tööle
- Kuva oma sisestatud sõna lauseliikmed
- Loe tekst sisse failist. Kuva teksti lauseliikmete puu

Algandmete näitamine puus

Puu aitab välja tuua andmete struktuuri, selle juures aga on õpetlik näha, et milliste andmete põhjal sarnane struktuur on tekkinud. Siin näites koostame puu sõnaliikidest, pärast aga vaatame iga haru juures, et milliste lausealguste järgi vastav koht puus tekkis. Et kõrvuti vaja nii sõnu endid kui sõnaliike, kasutame andmete hoidmisel Pandase dataframe, millena suudab estnltk Text-klass väärtused välja anda. Ploki võtmeks endiselt vastaval kohal olev sõnaliik, puus alanevad elemendid nende sõnaliikide võtmetega nagu ennegi, ploki metaanded aga alamelemendis nimega meta, sealseteks väärtusteks kogus ja sisu. Esimene loendab puu selle kohani jõudnud lausete arvu, teine aga on massiiv, kuhu pannakse vastavad lausealgused. Andmepuu väljatrükk näha pärast programmikoodi.

Hiljem trükitakse andmepuu välja HTMLina. Iga sõlm ul (unordered list) elemendina, seesolevad sõnaliigid li (list item) elemendina. Kuna puu sügavus sõltub lausete pikkustest ning koodiga ei ole piiratud, siis võtme väärtuse (sõnaliigi) väljatrüki järel kutsutakse kuva-funktsioon uuesti välja võimalike alamelementide sisu näitamiseks.

```
from estnltk import Text
laused=["Juku tuli kooli", "Juku tuli joostes kooli", "Kati jooksis koju"]
andmed={}
for lause in laused:
  plokk=andmed
  df=Text(lause).get.word_texts.postags.as_dataframe
  lausealgus=""
  for reanr in range(len(df)):
    sonaliik=df["postags"][reanr]
    lausealgus+=df["word_texts"][reanr]+" "
    if sonaliik in plokk:
      plokk[sonaliik]['meta']['kogus']+=1
      plokk[sonaliik]['meta']['sisu'].append(lausealgus)
    else:
      plokk[sonaliik]={
                 'meta':{
                   'kogus': 1,
                   'sisu':[lausealgus]
                 }
      }
    plokk=plokk[sonaliik]

print(andmed)
f=open("vastus2.html", "w")
f.write("<html><body><h1>Andmepuu</h1>")

def kuva(solm):
   f.write("<ul>")
   for voti in solm:
     if not voti=="meta":
       f.write("<li>"+voti+"</li>")
       kuva(solm[voti])
   f.write("</ul>")

kuva(andmed)

f.write("</body></html>")
f.close()
```

Tekstiekraani väljund

```
jaagup@praktika1 ~/public_html/2018/kval/21puuandmed $ python3.5 puu2.py
 {'H': {'meta': {'kogus': 3, 'sisu': ['Juku ', 'Juku ', 'Kati ']}, 'V': {'S': {'meta':
{'kogus': 2, 'sisu': ['Juku tuli kooli ', 'Kati jooksis koju ']}}, 'meta': {'kogus': 3,
'sisu': ['Juku tuli ', 'Juku tuli ', 'Kati jooksis ']}, 'V': {'S': {'meta': {'kogus': 1,
'sisu': ['Juku tuli joostes kooli ']}}, 'meta': {'kogus': 1, 'sisu': ['Juku tuli joostes
']}}}}}
```

Sama andmestruktuuri esitus trepituna, meta-elemendid on arusaadavuse mõttes iga elemendi puhul esimeseks tõstetud. Pärisnimega (H) algab kolm lauset. Sealse

meta-elemendi juures on seetõttu koguseks 3 ning massiivis näha, milliste nimedega laused hakkasid. Praegusel juhul kõikide lausete puhul järgneb nimisõnale tegusõna (V), näha ka vastavad algused. Kolmandaks elemendiks on kahel korral nimisõna (S) koos vastavate lausealgustega, ühel korral tegusõna, millele omakorda järgneb nimisõna - nii need ka andmepuus nähtavad.

```
{'H': {'meta': {'kogus': 3, 'sisu': ['Juku ', 'Juku ', 'Kati ']},
  'V': { 'meta': {'kogus': 3, 'sisu': ['Juku tuli ', 'Juku tuli ', 'Kati jooksis ']},
    'S': {'meta': {'kogus': 2, 'sisu': ['Juku tuli kooli ', 'Kati jooksis koju ']}},
    'V': {'meta': {'kogus': 1, 'sisu': ['Juku tuli joostes ']},
      'S': {'meta': {'kogus': 1, 'sisu': ['Juku tuli joostes kooli ']}}
     }
  }
 }
}
```

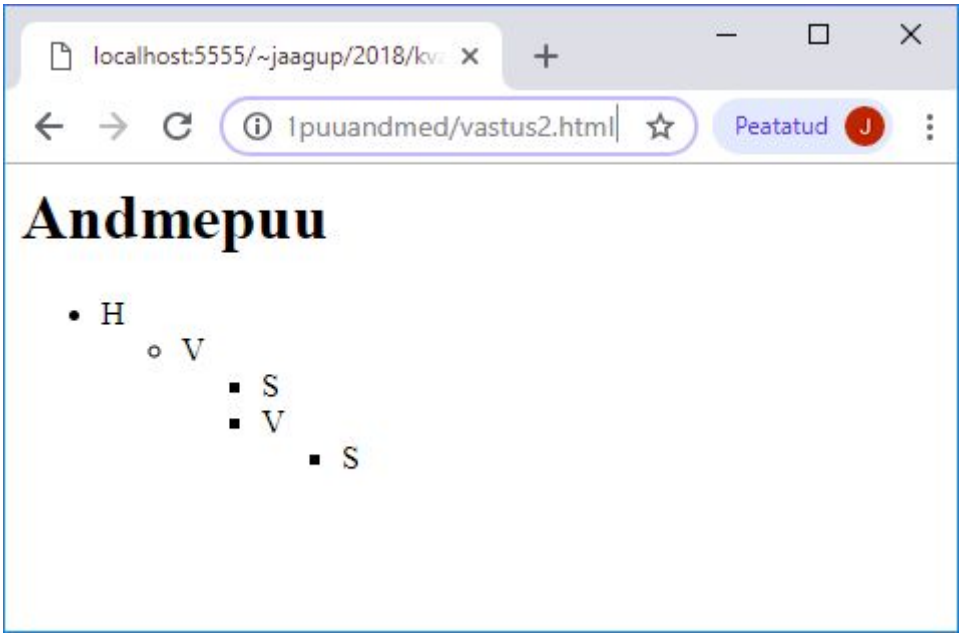
HTML-i väljund faili

```
<html><body><h1>Andmepuu</h1><ul><li>H</li><ul><li>V</li><ul><li>S</li><ul></ul><li>V</li><
ul><li>S</li><ul></ul></ul></ul></ul></ul></body></html>
```

Sama tekst trepituna. Rudimendina on sees tühjad ul-elemendid lause viimaste sõnade puhul, nendele aga leiame järgmises näites parema lahenduse

```
<html>
 <body>
  <h1>Andmepuu</h1>
  <ul>
   <li>H</li>
   <ul>
    <li>V</li>
    <ul>
     <li>S</li>
     <ul></ul>
     <li>V</li>
     <ul>
       <li>S</li>
       <ul></ul>
     </ul>
    </ul>
   </ul>
  </ul>
 </body>
</html>
```

Tulemus veebilehitseja ekraanil



Täiendused uues versioonis.

Aktiivsest sõlmest võetakse kõik alamelemendid peale meta, et saada vastavalt kohalt jätkuvad sõnaliigid. Neist luuakse massiiv, kus esimeseks elemendiks on vastaval kohal selle sõnaliigiga lausete arv ning teisel sõnaliik ise. Käsklus sorted järjestab massiividena olevad elemendid esimese elemendi (ehk praegusel juhul koguse) järgi kasvavas järjekorras, kui pole eraldi määratud teisiti. Kuna soovime levinumaid järgnevusi ettepoole, siis pöörame järjekorra reverse-käskluse abil ümber

```
jarjestatud=sorted([[solm[voti]["meta"]["kogus"], voti]
                        for voti in solm if not voti=="meta"])
jarjestatud.reverse()
```

Uuest järjestatud järjestatud loetelust küsime välja võtmed (ehk andmepaari elemendi kohal 1, kohal 0 on kogus). Selle võtme järgi saame meta-ploki sisu elemendist kätte massiivi sinna ulatuvate lausetega, mille join-käskluse abil järjest ridadele pandud tekstiks sätime.

```
for voti in [paar[1] for paar in jarjestatud]:
    f.write("<li title='"+("\n".join(solm[voti]["meta"]["sisu"]))+"'>"+voti+
       " ("+str(solm[voti]["meta"]["kogus"])+")</li>")
```

Kui sõlmel on alamelemente (peale meta), siis kutsume kuvamisfunktsiooni uuesti välja

```
    if len(solm[voti])>1: kuva(solm[voti])
```

Kood tervikuna

```
from estnltk import Text
laused=["Sinine vagun","Juku tuli kooli", "rõõmus juku tuli joostes kooli",
  "Kati jooksis koju", "Istub kodus", "Jookseb õues", "ka", "Katrin"]
andmed={}
for lause in laused:
```

```
  plokk=andmed
  df=Text(lause).get.word_texts.postags.as_dataframe
  lausealgus=""
  for reanr in range(len(df)):
    sonaliik=df["postags"][reanr]
    lausealgus+=df["word_texts"][reanr]+" "
    if sonaliik in plokk:
      plokk[sonaliik]['meta']['kogus']+=1
      plokk[sonaliik]['meta']['sisu'].append(lausealgus)
    else:
      plokk[sonaliik]={
                'meta':{
                  'kogus': 1,
                  'sisu':[lausealgus]
                }
      }
    plokk=plokk[sonaliik]

print(andmed)
f=open("vastus5.html", "w")
f.write("<html><body><h1>Andmepuu</h1>")

def kuva(solm):
   f.write("<ul>")
   jarjestatud=sorted([[solm[voti]["meta"]["kogus"], voti]
     for voti in solm if not voti=="meta"])
   jarjestatud.reverse()
   for voti in [paar[1] for paar in jarjestatud]:
      if solm[voti]["meta"]["kogus"]>0:
       f.write("<li title='"+("\n".join(solm[voti]["meta"]["sisu"]))+"'>"+voti+
          " ("+str(solm[voti]["meta"]["kogus"])+")</li>")
       if len(solm[voti])>1: kuva(solm[voti])
   f.write("</ul>\n")

kuva(andmed)

f.write("</body></html>")
f.close()
```

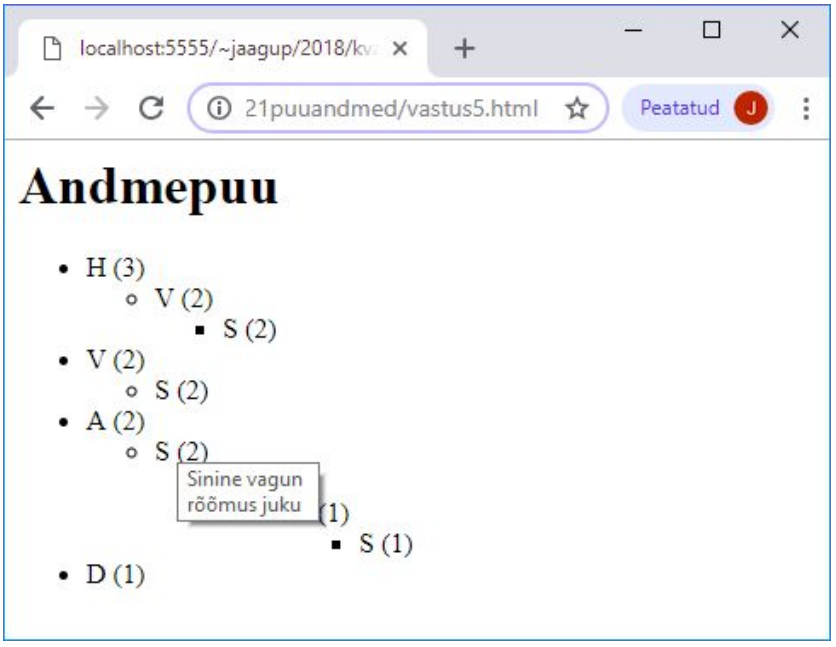
HTML-väljund

```
<html><body><h1>Andmepuu</h1><ul><li title='Juku
Kati
Katrin '>H (3)</li><ul><li title='Juku tuli
Kati jooksis '>V (2)</li><ul><li title='Juku tuli kooli
Kati jooksis koju '>S (2)</li></ul>
</ul>
<li title='Istub
Jookseb '>V (2)</li><ul><li title='Istub kodus
Jookseb õues '>S (2)</li></ul>
<li title='Sinine
rõõmus '>A (2)</li><ul><li title='Sinine vagun
rõõmus juku '>S (2)</li><ul><li title='rõõmus juku tuli '>V (1)</li><ul><li title='rõõmus
juku tuli joostes '>V (1)</li><ul><li title='rõõmus juku tuli joostes kooli '>S
(1)</li></ul>
</ul>
</ul>
</ul>
<li title='ka '>D (1)</li></ul>
</body></html>
```

Sama trepituna. Lühiduse mõttes reavahetused lauselõikude vahelt välja võetud, ehkki nõnda ei teki nad lausete vahele ka veebilehitsejas

```
<html>
 <body>
  <h1>Andmepuu</h1>
  <ul>
    <li title='Juku Kati Katrin '>H (3)</li>
    <ul>
      <li title='Juku tuli Kati jooksis '>V (2)</li>
      <ul>
        <li title='Juku tuli kooli Kati jooksis koju '>S (2)</li>
      </ul>
    </ul>
    <li title='Istub  Jookseb '>V (2)</li>
    <ul>
      <li title='Istub kodus Jookseb õues '>S (2)</li>
    </ul>
    <li title='Sinine rõõmus '>A (2)</li>
    <ul>
      <li title='Sinine vagun rõõmus juku '>S (2)</li>
      <ul>
        <li title='rõõmus juku tuli '>V (1)</li>
        <ul>
          <li title='rõõmus juku tuli joostes '>V (1)</li>
           <ul><li title='rõõmus juku tuli joostes kooli '>S (1)</li></ul>
        </ul>
      </ul>
    </ul>
    <li title='ka '>D (1)</li>
   </ul>
  </body>
</html>
```

Tulemus brauseris. Kui minna hiirega puu elemendi kohale, siis näeb seal, milliste lausealguste abil vastavasse kohta jõuti

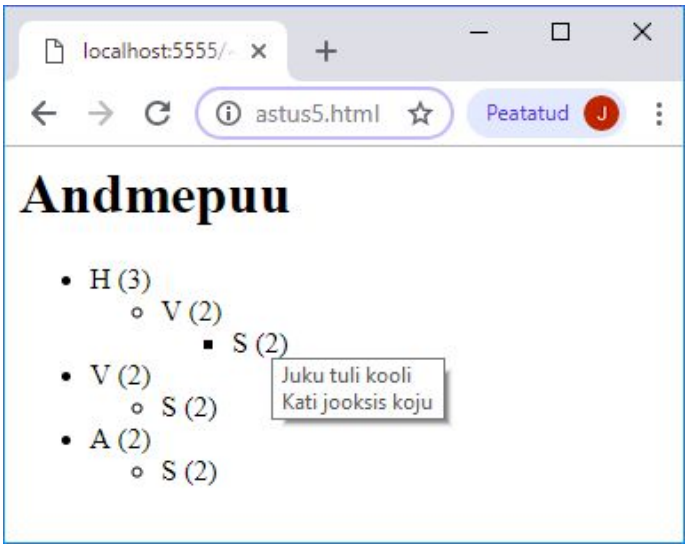


Näites piirdusime puu loomise juures paari lausega. Tekstides mustrite leidmiseks kasutatakse aga pigem sadu ja tuhandeid lauseid, vahel isegi kümneid tuhandeid sõnu. Neist ligi pooled järgnevused jäävad ühekordseteks, ülejäänust pool mõnekordseteks ning ka suure andmestiku korral koorub üldjuhul lõpuks välja hallatava suurusega tuumikpuu, milles leiduvad mustrid näitavad välja enam levinud järgnevused.

Puu suuruse piiramiseks lisasime rasvases kirjas märgitud tingimuslause - puu harusse minnakse kuvamiseks sisse vaid siis, kui märgitud kogus on piisavalt suur, praegusel juhul suurem kui üks.

```
def kuva(solm):
   f.write("<ul>")
   jarjestatud=sorted([[solm[voti]["meta"]["kogus"], voti]
     for voti in solm if not voti=="meta"])
   jarjestatud.reverse()
   for voti in [paar[1] for paar in jarjestatud]:
      if solm[voti]["meta"]["kogus"]>1:
       f.write("<li title='"+("\n".join(solm[voti]["meta"]["sisu"]))+"'>"+voti+
          " ("+str(solm[voti]["meta"]["kogus"])+")</li>")
       if len(solm[voti])>1: kuva(solm[voti])
   f.write("</ul>\n")
```

Kuvatav puu tuleb niimoodi mõnevõrra väiksem, kergemini vaadatav ning näitlaused endiselt sees näha.



Harjutus

- Pane näited käima
- Kuva sõnade tähtedest sageduste järgi järjestatud puu

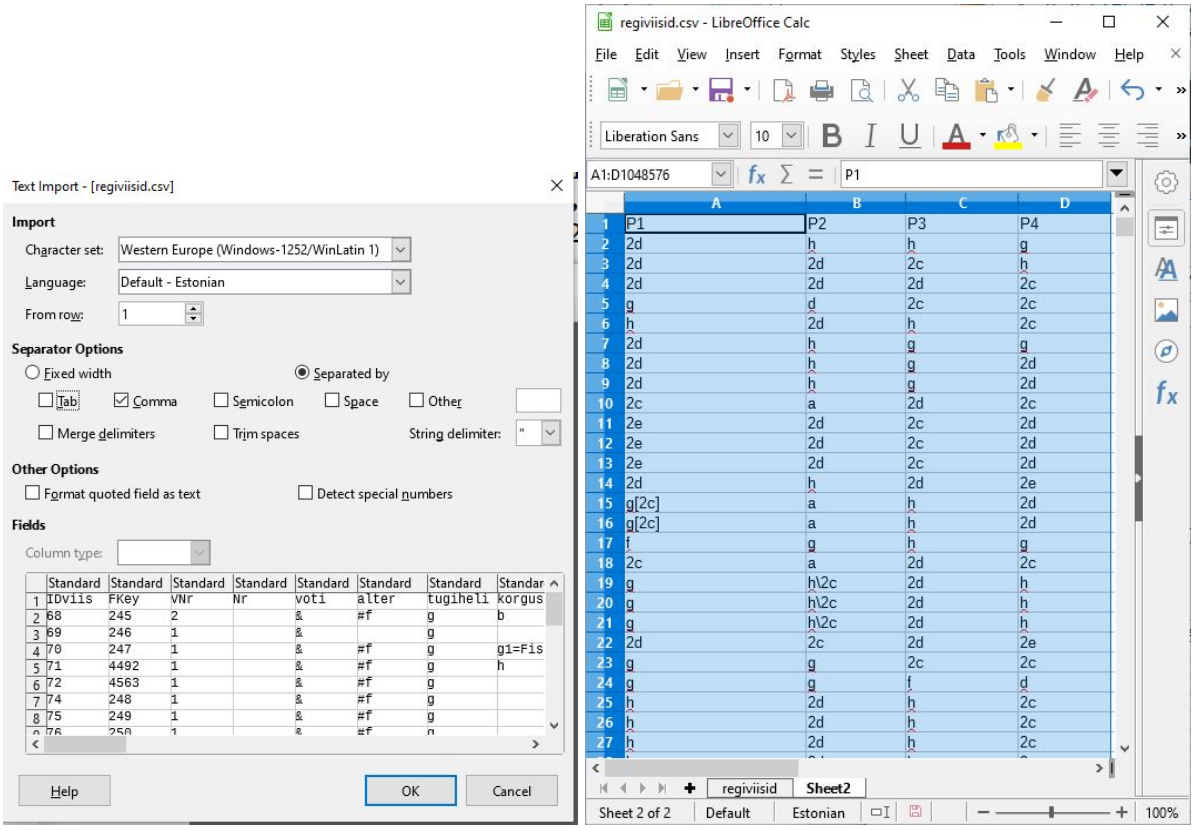
Andmepuu regiviiside näitel

Eesti Kirjandusmuuseumi Eesti Rahvaluule Arhiivis on talletatud hulgaliselt regilaulude viise. Andmebaasi meloodiatest väljavõtte leiab aadressilt

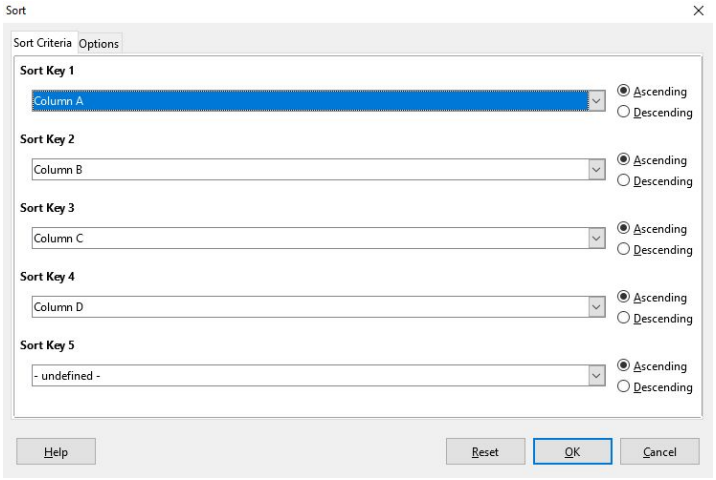
`http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt`

Tabelarvutusleht

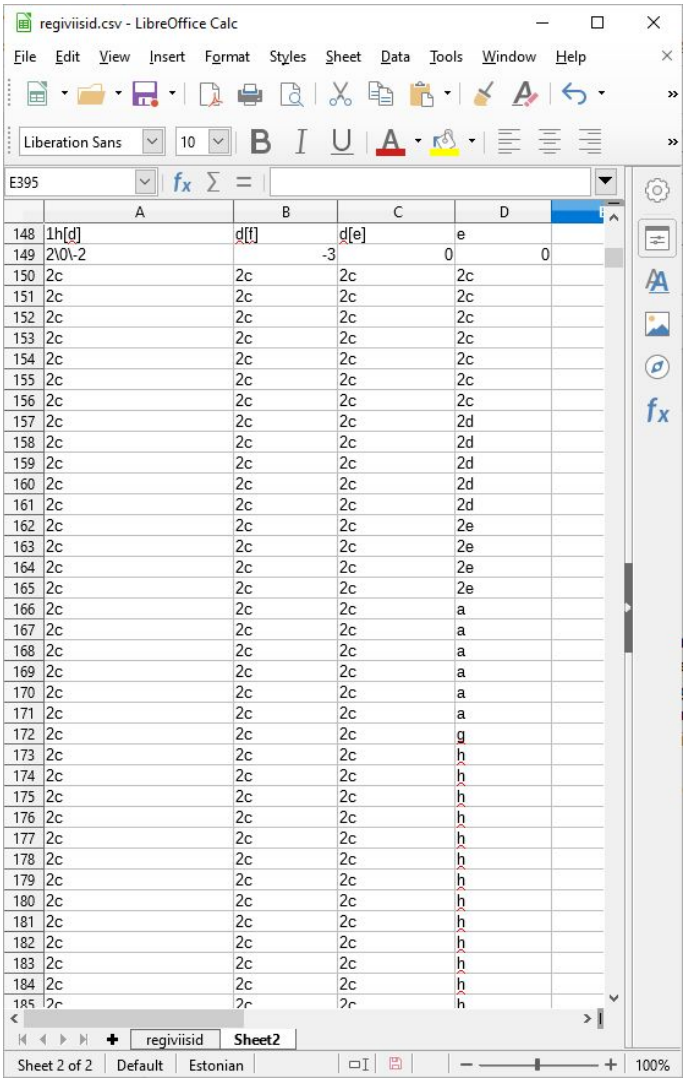
Esmaseks uurimiseks sobib see importida tabelarvutussüsteemi - näiteks Libre Office Calc programmi. Tulpade eraldajaks koma.



Kui sisse võtmine läks edukalt, jõuavad andmed tabelisse. Veidi uurides paistab, et viis ise algab tulbast pealkirjaga P1 - lihtsamal juhul eemaldame eelnevad tulbad ning järjestame viisid tähestikuliselt



Tulemuseks koonduvad samasuguse algusega viisid kokku. Algusesse jäid mitmesugused eripärased märgistused, pärast sajandat rida hakkavad aga tavalised noodid - praeguses kohas teise oktaavi C-st. Esimesed kuus viisi algavad kolme 2c-ga, järgmise nelja puhul on algusnootideks 2c-2c-2d. Nõnda näeb väiksemate koguste puhul silmaga ära, et millised algused levinumad on.



	A	B	C	D
148	1h[d]	d[f]	d[e]	e
149	2\0\-2	-3	0	0
150	2c	2c	2c	2c
151	2c	2c	2c	2c
152	2c	2c	2c	2c
153	2c	2c	2c	2c
154	2c	2c	2c	2c
155	2c	2c	2c	2c
156	2c	2c	2c	2c
157	2c	2c	2c	2d
158	2c	2c	2c	2d
159	2c	2c	2c	2d
160	2c	2c	2c	2d
161	2c	2c	2c	2d
162	2c	2c	2c	2e
163	2c	2c	2c	2e
164	2c	2c	2c	2e
165	2c	2c	2c	2e
166	2c	2c	2c	a
167	2c	2c	2c	a
168	2c	2c	2c	a
169	2c	2c	2c	a
170	2c	2c	2c	a
171	2c	2c	2c	a
172	2c	2c	2c	g
173	2c	2c	2c	h
174	2c	2c	2c	h
175	2c	2c	2c	h
176	2c	2c	2c	h
177	2c	2c	2c	h
178	2c	2c	2c	h
179	2c	2c	2c	h
180	2c	2c	2c	h
181	2c	2c	2c	h
182	2c	2c	2c	h
183	2c	2c	2c	h
184	2c	2c	2c	h
185	2c	2c	2c	h

Ligipääs programmeerimiskeelega

Viisidele pääseb ligi ka programmeerimiskeele abil. Kui Pythoni kood käivitub andmefailiga samas kataloogis, siis saab faili selle nime järgi avada. Esimene rida on pealkirjade rida - selle loeme ja jätame praeguse seisuga kasutamata. Edasi saab juba soovi järgi konkreetseid väärtusi lugeda ja kasutada. Käsklus `split` tükeldab teksti lõikudeks, praegusel juhul koma koha pealt. Viisi algustulp (P1) on üheteistkümnes, Pythoni moodi nullist lugedes järjekorranumbriga kümme. Nii saabki esimese viisi esimese noodi (2d) kätte.

```
f=open("regiviisid.txt")
f.readline()
print(f.readline().split(",")[10])
```

Käivitus

```
$ python3.5 noodid1.py
2d
```

Andmemajanduseks on Pythonil pandas-teek. Selle käsklus `read_csv` võimaldab komadega eraldatud tabeli tervikuna sisse lugeda. Edasi on võimalik juba tulba nime järgi pöörduda üksikute väärtuste poole tabelis. Kui anda ette ka rea järjekorranumber, siis saab kätte üksiku väärtuse (2d nagu eelmises näites), muul juhul tuleb välja terve tulp korraga.

```
import pandas as pd
df=pd.read_csv("regiviisid.txt")
#print(df["P1"][0])
print(df["P1"])
```

Käivitus näitab välja esimeste viiside avanoodid.

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/0904 $ python3.5
noodid2.py
0          2d
1          2d
2          2d
3           g
4           h
```

Esimeste erinevate nootide sageduste kokku lugemiseks sobib pandase dataframe tulba käsklus `value_counts`

```
import pandas as pd
df=pd.read_csv("regiviisid.txt")
print(df["P1"].value_counts())
```

Nii näeb, et enim lugusid algab g-noodiga, aga leidub ka muid salapäraseid kirjapanekumooduseid.

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/0904 $ python3.5
noodid2.py
g                           1151
2d                           827
h                            674
2c                           564
a                            560
2e                           203
e                            189
f                            144
c                            101
d                             97
kodeeritult kuskil olemas     88
-3                            31
```

Tagasi "tavalise" Pythoni juurde. Tulba käik väärtused saab kätte tsükli abil

```
f=open("regiviisid.txt")
f.readline()
for rida in f:
  print(rida.split(",")[10])
```

Võti `more` käivitamise juures hoolitseb, et noodihulk pärast ekraani täitmist me silmade ette pidama jääb ning alles klahvivajutusega on võimalik edasi liikuda.

```
python3.5 noodid3.py | more
2d
2d
2d
g
h
2d
```

Väärtuste loendamine

Alustuseks ridade arvu lugemine. Muutuja nimeks puu selle mõttega, et hiljem võiks sellest andmepuu välja kasvada ning puu juurelemendina loendame kõikide viiside arvu.

```
puu={"kogus":0}
f=open("regiviisid.txt")
f.readline()
for rida in f:
  puu["kogus"]+=1

print(puu["kogus"])
```

Vastus

```
4941
```

Edasi teeme iga alamelemendi ehk loo esimese noodi väärtuse kohta koguse, et mitu korda see esineb. Kui vastavat nooti veel puu juure küljes pole, siis lisatakse see sinna kogusega 1, kui juba olemas, siis suurendatakse kogust ühe võrra.

```
puu={"kogus":0}
f=open("regiviisid.txt")
f.readline()
for rida in f:
  puu["kogus"]+=1
  noot1=rida.split(",")[10]
  if noot1 not in puu:
     puu[noot1]={"kogus":1}
  else:
     puu[noot1]["kogus"]+=1

print(puu["kogus"])
print(puu["g"]["kogus"])

print(puu)
```

Käivitamisel paistab, et kogu puus on 4941 viisi, millest g-noodiga algab 1151. Edasi järgneb üksikute algusnootide koguste loetelu. Nagu näha, siis 2 oktaavi d-ga algab 827 lugu, järgmiseks elemendiks on üldkogus ning siis tulevad üksikute algusnootide kogused riburadapidi järele

```
python3.5 noodid3.py
```

```
4941
1151
{'2d': {'kogus': 827}, 'kogus': 4941, '2c[a]': {'kogus': 3}, 'e[g]': {'kogus': 1},
'[2]]-3]': {'kogus': 1}, '2d[2f]': {'kogus': 1}, '[0][-3]': {'kogus': 1}, '2g': {'kogus':
4}, 'g[h]': {'kogus': 23}, '2': {'kogus': 6}, '[-5][0]': {'kogus': 1}, '1g': {'kogus': 1},
'3': {'kogus': 18}, '1h[d]': {'kogus': 1}, 'a\\2d': {'kogus': 1}, '(2c)h': {'kogus': 1},
'd[c]': {'kogus': 4}, 'h[g]': {'kogus': 9}, '3\\1': {'kogus': 2}, 'e[d\\e][f]': {'kogus':
1}, 'a': {'kogus': 560}, '1a': {'kogus': 2}, 'e\\d\\d': {'kogus': 1}, 'g': {'kogus': 1151},
'h\\2c': {'kogus': 2}, 'h[a]': {'kogus': 2}, '#2c': {'kogus': 1}, '4': {'kogus': 3},
'[-4][-2]': {'kogus': 1}, '[-3][3]': {'kogus': 1}, 'e[e/d]': {'kogus': 1}, 'g[d]':
{'kogus': 3}, '-1\\-3': {'kogus': 2}, 'e': {'kogus': 189}, 'h': {'kogus': 674}, '3e':
{'kogus': 1}, 'd': {'kogus': 97}, '-3\\-5': {'kogus': 1}, '2d[2c]': {'kogus': 4}, '2e':
{'kogus': 203}, '2d\\2c': {'kogus': 1}, 'g[f]': {'kogus': 1}, '2c\\2d': {'kogus': 1}, '-4':
{'kogus': 23}, 'c[e]': {'kogus': 1}, 'h/a': {'kogus': 1}, 'f[d]': {'kogus': 1}, 'g[2c]':
{'kogus': 2}, 'a[a\\2c]': {'kogus': 1}, 'kus viis hulgub?': {'kogus': 12}, '2c[h]':
{'kogus': 2}, 'kodeeritult kuskil olemas': {'kogus': 88}, '3\\2': {'kogus': 1}, 'otsi
orig-st': {'kogus': 1}, 'a\\2c': {'kogus': 3}, '[-2][-3]': {'kogus': 1}, '2d\\h[h]':
{'kogus': 4}, '???': {'kogus': 1}, 'h[h/g]': {'kogus': 1}, '2f': {'kogus': 5}, 'c[d]':
{'kogus': 1}, '[3][-3]': {'kogus': 1}, '-1': {'kogus': 3}, '2c[2e]': {'kogus': 1}, 'h[2c]':
{'kogus': 2}, '2d[h]': {'kogus': 6}, 'g[2d]': {'kogus': 29}, 'h/2d': {'kogus': 1},
'[-5][3': {'kogus': 1}, '-2\\-3': {'kogus': 2}, 'e\\c': {'kogus': 1}, '[-4][3]': {'kogus':
2}, '2c': {'kogus': 564}, '2c\\h': {'kogus': 2}, 'f': {'kogus': 144}, '2d[g]': {'kogus':
7}, 'd[1h]': {'kogus': 2}, '-3': {'kogus': 31}, 'a[e][g]': {'kogus': 3}, 'h\\a': {'kogus':
2}, '-5': {'kogus': 8}, 'f[f\\f]': {'kogus': 1}, '1h': {'kogus': 6}, '0': {'kogus': 16},
'e[d]': {'kogus': 1}, 'g[g\\a]': {'kogus': 1}, 'g[a]': {'kogus': 4}, 'a[h]': {'kogus': 4},
'd\\f': {'kogus': 1}, 'h\\g': {'kogus': 1}, 'f\\d': {'kogus': 1}, 'h\\2d': {'kogus': 2},
'2d\\h': {'kogus': 1}, 'c': {'kogus': 101}, 'h[2e]': {'kogus': 4}, 'g\\h': {'kogus': 1},
'h[f]': {'kogus': 1}, 'a\\h': {'kogus': 4}, '-2': {'kogus': 11}, 'f[g]': {'kogus': 4},
'[-3][-4]': {'kogus': 1}, 'd\\e': {'kogus': 1}, 'a[f]': {'kogus': 2}, 'g\\a': {'kogus': 1},
'a[a\\h]': {'kogus': 1}, '2\\0\\-2': {'kogus': 1}}
```

Esialgsest segamini järjekorras andmestikust on tulemusi tülikas välja lugeda. Sortides algusnoodid nende esinemise koguste järgi, leiab haruldasemad ja levinumad alguskohad kergemini kätte. Nii võetakse puu juurest välja kõik noodid (kogus on erinimi, mis jäetakse välja) ning tehakse andmepaarid (kaheelemendilised massiivid), kus esimeseks väärtuseks kogus ja teiseks noodi nimi. Käsklus sorted järjestab massiivi esimese elemendi järgi, nii tulevad andmed koguste järgi ritta.

```
print(list(sorted([puu[noot]["kogus"], noot] for noot in puu if not noot=="kogus")))
```

Nagu näha, siis lõpus on suuremad kogused

```
[[1, '#2c'], [1, '(2c)h'], [1, '-3\\-5'], [1, '1g'], [1, '1h[d]'], [1, '2\\0\\-2'], [1,
'2c[2e]'], [1, '2c\\2d'], [1, '2d[2f]'], [1, '2d\\2c'], [1, '2d\\h'], [1, '3\\2'], [1,
'3e'], [1, '???'], [1, '[-2][-3]'], [1, '[-3][-4]'], [1, '[-3][3]'], [1, '[-4][-2]'], [1,
'[-5][0]'], [1, '[-5][3'], [1, '[0][-3]'], [1, '[2]]-3]'], [1, '[3][-3]'], [1, 'a[a\\2c]'],
[1, 'a[a\\h]'], [1, 'a\\2d'], [1, 'c[d]'], [1, 'c[e]'], [1, 'd\\e'], [1, 'd\\f'], [1,
'e[d\\e][f]'], [1, 'e[d]'], [1, 'e[e/d]'], [1, 'e[g]'], [1, 'e\\c'], [1, 'e\\d\\d'], [1,
'f[d]'], [1, 'f[f\\f]'], [1, 'f\\d'], [1, 'g[f]'], [1, 'g[g\\a]'], [1, 'g\\a'], [1,
'g\\h'], [1, 'h/2d'], [1, 'h/a'], [1, 'h[f]'], [1, 'h[h/g]'], [1, 'h\\g'], [1, 'otsi
orig-st'], [2, '-1\\-3'], [2, '-2\\-3'], [2, '1a'], [2, '2c[h]'], [2, '2c\\h'], [2,
'3\\1'], [2, '[-4][3]'], [2, 'a[f]'], [2, 'd[1h]'], [2, 'g[2c]'], [2, 'h[2c]'], [2,
'h[a]'], [2, 'h\\2c'], [2, 'h\\2d'], [2, 'h\\a'], [3, '-1'], [3, '2c[a]'], [3, '4'], [3,
'a[e][g]'], [3, 'a\\2c'], [3, 'g[d]'], [4, '2d[2c]'], [4, '2d\\h[h]'], [4, '2g'], [4,
'a[h]'], [4, 'a\\h'], [4, 'd[c]'], [4, 'f[g]'], [4, 'g[a]'], [4, 'h[2e]'], [5, '2f'], [6,
'1h'], [6, '2'], [6, '2d[h]'], [7, '2d[g]'], [8, '-5'], [9, 'h[g]'], [11, '-2'], [12, 'kus
viis hulgub?'], [16, '0'], [18, '3'], [23, '-4'], [23, 'g[h]'], [29, 'g[2d]'], [31, '-3'],
```

```
[88, 'kodeeritult kuskil olemas'], [97, 'd'], [101, 'c'], [144, 'f'], [189, 'e'], [203,
'2e'], [560, 'a'], [564, '2c'], [674, 'h'], [827, '2d'], [1151, 'g']]
```

Parameeter `reverse=True` keerab sorteerimisjärjestuse ümber

```
print(list(sorted(([puu[noot]["kogus"], noot] for noot in puu if not noot=="kogus"),
reverse=True)))
```

loetelus nõnda suuremad sagedused ees

```
[[1151, 'g'], [827, '2d'], [674, 'h'], [564, '2c'], [560, 'a'], [203, '2e'], [189, 'e'],
[144, 'f'], [101, 'c'], [97, 'd'], [88, 'kodeeritult kuskil olemas'], [31, '-3'], [29,
'g[2d]'], [23, 'g[h]'], [23, '-4'], [18, '3'], [16, '0'], [12, 'kus viis hulgub?'], [11,
'-2'], [9, 'h[g]'], [8, '-5'], [7, '2d[g]'], [6, '2d[h]'], [6, '2'], [6, '1h'], [5, '2f'],
[4, 'h[2e]'], [4, 'g[a]'], [4, 'f[g]'], [4, 'd[c]'], [4, 'a\\h'], [4, 'a[h]'], [4, '2g'],
[4, '2d\\h[h]'], [4, '2d[2c]'], [3, 'g[d]'], [3, 'a\\2c'], [3, 'a[e][g]'], [3, '4'], [3,
'2c[a]'], [3, '-1'], [2, 'h\\a'], [2, 'h\\2d'], [2, 'h\\2c'], [2, 'h[a]'], [2, 'h[2c]'],
[2, 'g[2c]'], [2, 'd[1h]'], [2, 'a[f]'], [2, '[-4][3]'], [2, '3\\1'], [2, '2c\\h'], [2,
'2c[h]'], [2, '1a'], [2, '-2\\-3'], [2, '-1\\-3'], [1, 'otsi orig-st'], [1, 'h\\g'], [1,
'h[h/g]'], [1, 'h[f]'], [1, 'h/a'], [1, 'h/2d'], [1, 'g\\h'], [1, 'g\\a'], [1, 'g[g\\a]'],
[1, 'g[f]'], [1, 'f\\d'], [1, 'f[f\\f]'], [1, 'f[d]'], [1, 'e\\d\\d'], [1, 'e\\c'], [1,
'e[g]'], [1, 'e[e/d]'], [1, 'e[d]'], [1, 'e[d\\e][f]'], [1, 'd\\f'], [1, 'd\\e'], [1,
'c[e]'], [1, 'c[d]'], [1, 'a\\2d'], [1, 'a[a\\h]'], [1, 'a[a\\2c]'], [1, '[3][-3]'], [1,
'[2]]-3]'], [1, '[0][-3]'], [1, '[-5][3'], [1, '[-5][0]'], [1, '[-4][-2]'], [1, '[-3][3]'],
[1, '[-3][-4]'], [1, '[-2][-3]'], [1, '???'], [1, '3e'], [1, '3\\2'], [1, '2d\\h'], [1,
'2d\\2c'], [1, '2d[2f]'], [1, '2c\\2d'], [1, '2c[2e]'], [1, '2\\0\\-2'], [1, '1h[d]'], [1,
'1g'], [1, '-3\\-5'], [1, '(2c)h'], [1, '#2c']]
```

Puhtamaks välja lugemiseks levinumad paarid

```
for paar in sorted(([puu[noot]["kogus"], noot] for noot in puu if not noot=="kogus"),
reverse=True):
  if paar[0]>100:
    print(paar[1], ' - ', paar[0])
```

Tekkinud loetelu

```
g  -  1151
2d  -  827
h  -  674
2c  -  564
a  -  560
2e  -  203
e  -  189
f  -  144
c  -  101
```

Harjutus

Koosta tekstifail mõne silbitatud sõnaga. Nt

ma-gus
ma-ga-ma

kap-sa-uss

- Loe programmi abil kokku, mitu korda millist esimest silpi on
- Kuva silbid koos sagedustega kahanevas järjekorras

```
f=open("silbid1.txt")
puu={"kogus":0}
for rida in f:
  puu["kogus"]+=1
  silp=rida.split("-")[0]
  if not silp in puu:
    puu[silp]={"kogus":1}
  else:
    puu[silp]["kogus"]+=1

print(puu)

for paar in sorted([[puu[silp]["kogus"], silp] for silp in puu if not silp=="kogus"],
reverse=True):
  print(paar[1], "-", paar[0])
```

Käivitus

```
python3.5 silbid1.py

{'kogus': 3, 'ma': {'kogus': 2}, 'kap': {'kogus': 1}}
ma - 2
kap - 1
```

Harjutus - puu koostamine

- Tutvu näitega alampeatüki "Puu loomine programmiga" lõpus. Käivita, koosta puid erisuguste sõnade tähtedega
- Võta sisendiks tekstifail silbitatud sõnadega. Koosta sõnasilpide puu. Kuva

Eripäraks et nüüd on juures muutuja aktiivse ploki meeles pidamiseks. Nii on võimalik puud koostada keskendudes korraga vaid ühele sõlmele ning selle juures andmete paika sättimisel liikuda edasi järgmise sõlme juurde. Uue sõna või muu üksuse analüüsil hakatakse aktiivse plokiga taas puu juurest peale ning liigutakse edasi senikaua, kuni sisendil väärtusi (praegusel juhul sõnal silpe) jagub

```
andmed={"kogus":0}
f=open("silbid1.txt")
for sona in f:
 plokk=andmed
 plokk["kogus"]+=1
 for silp in sona.strip().split("-"):
  if silp in plokk:
   plokk[silp]["kogus"]+=1
  else:
   plokk[silp]={"kogus":1}
```

```
  plokk=plokk[silp]

print(andmed)
```

Sisendfail

```
silbid.txt

ma-gus
ma-ga-ma
kap-sa-uss
```

väljund

```
python tahed1.py
{'kogus': 3, 'kap': {'kogus': 1, 'sa': {'kogus': 1, 'uss': {'kogus': 1}}}, 'ma': {'kogus':
2, 'gus': {'kogus': 1}, 'ga': {'kogus': 1, 'ma': {'kogus': 1}}}}
```

Sama viisipuuga

```
andmed={"kogus":0}
f=open("regiviisid.txt")
f.readline()
for rida in f:
 plokk=andmed
 plokk["kogus"]+=1
 for noot in rida.strip().split(",")[10:13]:
  if noot in plokk:
   plokk[noot]["kogus"]+=1
  else:
   plokk[noot]={"kogus":1}
  plokk=plokk[noot]

print(andmed)
```

Puu tuli nõnda päris suur, näha vaid väike lõik sellest

```
python viisipuu.py
{'2c\\2d': {'kogus': 1, '2e': {'kogus': 1, '2e': {'kogus': 1}}}, '[0][-3]': {'kogus': 1,
'[0][-3': {'kogus': 1, '0': {'kogus': 1}}}, '1h[d]': {'kogus': 1, 'd[f]': {'kogus': 1,
'd[e]': {'kogus': 1}}},
```

Andmete kätte saamine

Kõigepealt puu juurtaseme noodid. Langjoon for-tsükli real selleks, et saaks käsu sees rida vahetada

```
minkogus=100
def kuva(plokk):
   for kogus, noot in sorted([[plokk[noot]["kogus"], noot] \
                              for noot in plokk if not noot=="kogus"], reverse=True):
    if kogus>=minkogus:
     print(noot, kogus)
```

```
kuva(andmed)
```

Tulemus

```
python viisipuu2.py
g 1151
2d 827
h 674
2c 564
a 560
2e 203
e 189
f 144
c 101
```

Andmepuu väljatrükk

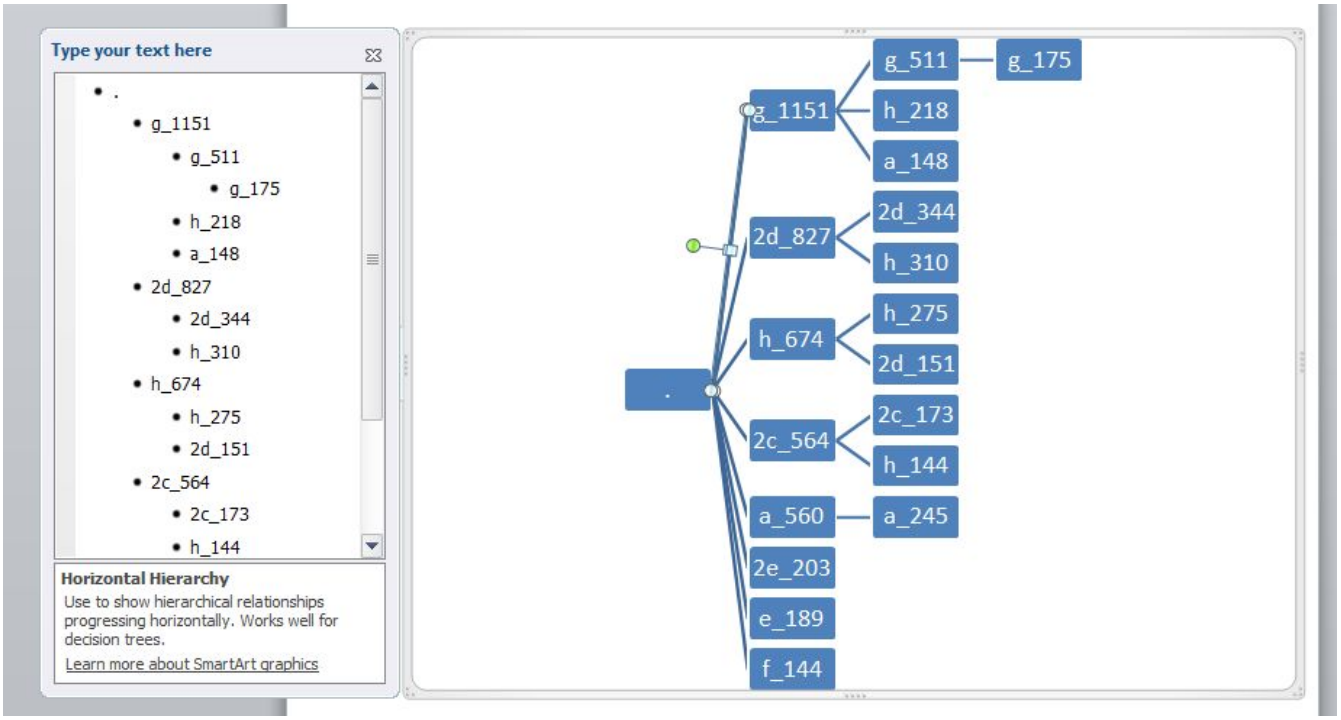
Siinses näites koostame puu viisi esimese kolme noodiga - tulbad 10-13 (viimane välja arvatud. Väljatrüki puhul juures pealtnäha väike, aga sisu poolest tähtis lisandus: lisaks noodi kuvamisel kutsutakse kuva-funktsioon uuest välja parajasti väljatrükitava noodi kohalt. Niimoodi paigutatakse vastava koha väljund ka sobivasse kohta. Tulemuse saadame faili, et vastus oleks loetav ka pikemate andmete puhul. Iga väljatrüki juures ühe tabulaatori võrra suurem taane - nii näha, et mis mille alla kuulub. Praeguses näites väljatrüki miinimumkoguseks 140 - nii püsib nootide/sõlmede arv silma ees veel jälgitav.

```
andmed={"kogus":0}
f=open("regiviisid.txt")
f.readline()
for rida in f:
 plokk=andmed
 plokk["kogus"]+=1
 for noot in rida.strip().split(",")[10:13]:
  if noot in plokk:
   plokk[noot]["kogus"]+=1
  else:
   plokk[noot]={"kogus":1}
  plokk=plokk[noot]
f.close()

minkogus=140
f2=open("vastus.txt", "w")
print(".\r", file=f2)
def kuva(plokk, taane):
   for kogus, noot in sorted([[plokk[noot]["kogus"], noot] \
                              for noot in plokk if not noot=="kogus"], reverse=True):
    if kogus>=minkogus:
      print(taane*"\t"+noot+"_"+str(kogus)+"\r", file=f2)
      kuva(plokk[noot], taane+1)

kuva(andmed, 1)
f2.close()
```

Tulemusfail, kus näha levinumad viisialgused. Näha noot ning selle esinemise sagedus.

```
vastus.txt
.
        g_1151
                g_511
                        g_175
                h_218
                a_148
        2d_827
                2d_344
                h_310
        h_674
                h_275
                2d_151
        2c_564
                2c_173
                h_144
        a_560
                a_245
        2e_203
        e_189
        f_144
```

Microsoft Wordi SmartArt võimaldab sellise puu kaudu joonise kuvada



Koodi ja andmeid veidi kohandades kuvame välja g-põhitooniga viiside g-ga algavad meloodialõigud - nii saab juba ühest alamosast parasjagu selgema pildi.

```
andmed={"kogus":0}
f=open("regiviisid.txt")
f.readline()
```

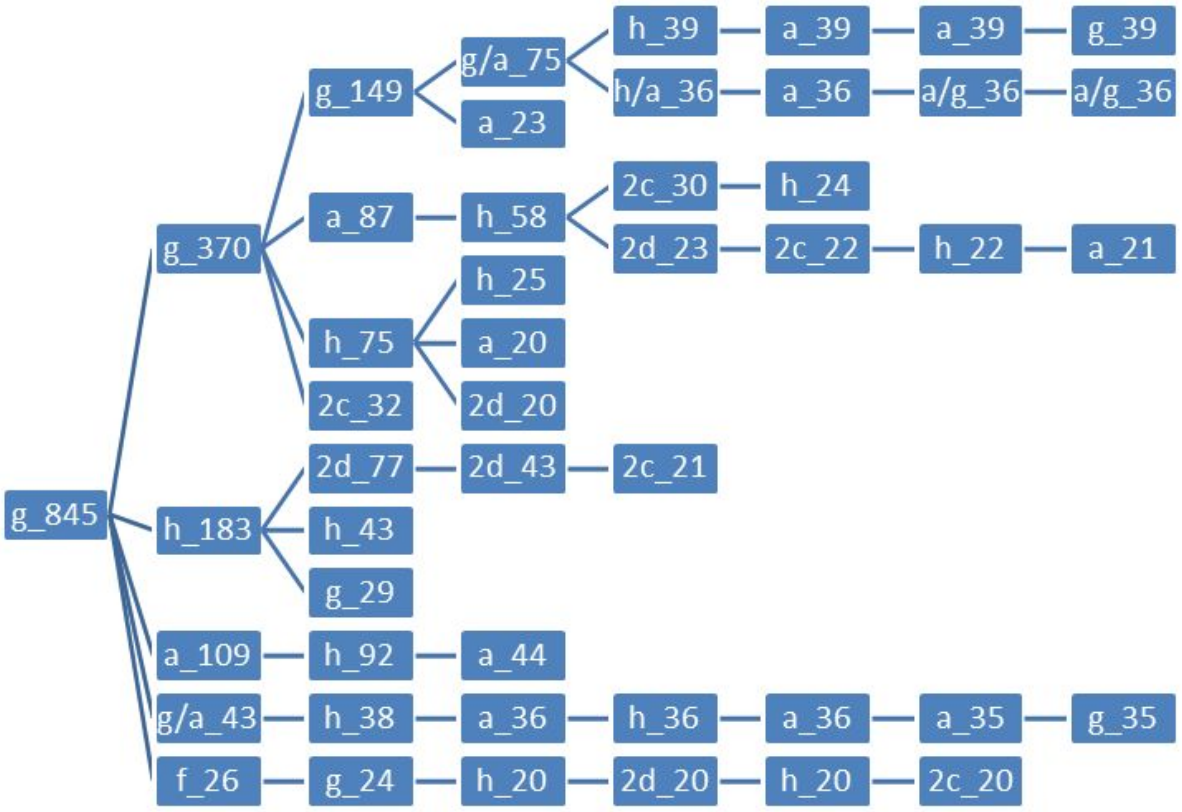
```
for rida in f:
 if rida.strip().split(",")[6]=='g':
  plokk=andmed
  plokk["kogus"]+=1
  for noot in rida.strip().split(",")[10:18]:
   if noot in plokk:
    plokk[noot]["kogus"]+=1
   else:
    plokk[noot]={"kogus":1}
   plokk=plokk[noot]
f.close()

minkogus=20
f2=open("vastus.txt", "w")
#print(".\r", file=f2)
def kuva(plokk, taane):
   for kogus, noot in sorted([[plokk[noot]["kogus"], noot] \
                              for noot in plokk if not noot=="kogus"], reverse=True):
    if kogus>=minkogus:
      print(taane*"\t"+noot+"_"+str(kogus)+"\r", file=f2)
      kuva(plokk[noot], taane+1)

kuva(andmed, 0)
f2.close()
```

Väljundfail, kus noodid nähtaval

```
g_845
       g_370
              g_149
                     g/a_75
                            h_39
                                   a_39
                                          a_39
                                                 g_39
                            h/a_36
                                   a_36
                                          a/g_36
                                                 a/g_36
                     a_23
              a_87
                     h_58
                            2c_30
                                   h_24
                            2d_23
                                   2c_22
                                          h_22
                                                 a_21
              h_75
                     h_25
                     a_20
                     2d_20
              2c_32
       h_183
              2d_77
                     2d_43
                            2c_21
              h_43
              g_29
       a_109
              h_92
```

```
                a_44
g/a_43
        h_38
                a_36
                        h_36
                                a_36
                                        a_35
                                                g_35
f_26
        g_24
                h_20
                        2d_20
                                h_20
                                        2c_20
```

Sama andmepuu põhjal koostatud SmartArt-i joonis



Harjutus

- Pane näide käima
- Kuva välja G-duuri 2d-ga algavate viiside puu

Järjestuste sagedused

Järjestuste leidmise kohta leiab pikema seletatud näite allolevast Digihumanitaaria tehnoloogiate konspektist peatükist **tähepaarid**

https://docs.google.com/document/d/1hsTX22h82cZqIynxdGgKkLdTI9aJYpTYuzmDXE5nl_Q

Selle näite põhjal leiame kahetähelised järjendid sõnast "allveesport"

```
sona="allveesport"
jadapikkus=2
jadad=[sona[arv:arv+jadapikkus] for arv in range(len(sona)-(jadapikkus-1))]
print(jadad)
```

tulemus

```
 python3.5 paarid1.py
['al', 'll', 'lv', 've', 'ee', 'es', 'sp', 'po', 'or', 'rt']
```

Sama koodilõik kolmetäheliste jadadega

```
sona="allveesport"
jadapikkus=3
jadad=[sona[arv:arv+jadapikkus] for arv in range(len(sona)-(jadapikkus-1))]
print(jadad)
```

tulemus

```
python3.5 paarid1.py
['all', 'llv', 'lve', 'vee', 'ees', 'esp', 'spo', 'por', 'ort']
```

Edasi näide mitme sõna juures jadade leidmiseks. Tsükliga käiakse sõnad läbi, iga sõna kohta leitakse jadad ning liidetakse üheks suureks jadaks kokku.

```
sonad=["allveesport", "sportlane", "meeskonnasport", "tugitoolisport"]
jadapikkus=3
koikjadad=[]
for sona in sonad:
    sonajadad=[sona[arv:arv+jadapikkus] for arv in range(len(sona)-(jadapikkus-1))]
    koikjadad+=sonajadad
print(koikjadad)
```

Tulemus

```
python3.5 paarid2.py
['all', 'llv', 'lve', 'vee', 'ees', 'esp', 'spo', 'por', 'ort', 'spo', 'por', 'ort', 'rtl',
'tla', 'lan', 'ane', 'mee', 'ees', 'esk', 'sko', 'kon', 'onn', 'nna', 'nas', 'asp', 'spo',
'por', 'ort', 'tug', 'ugi', 'git', 'ito', 'too', 'ool', 'oli', 'lis', 'isp', 'spo', 'por',
'ort']
```

Samale koodile juurde analüüsiosa. Klass nimega Counter võimaldab loetelus leiduvate eri väärtuste sagedused kokku lüüa. Tulemusena näha, et "ort", "spo" ja "por" tõusevad tugevasti esile.

```
sonad=["allveesport", "sportlane", "meeskonnasport", "tugitoolisport"]
jadapikkus=3
koikjadad=[]
for sona in sonad:
    sonajadad=[sona[arv:arv+jadapikkus] for arv in range(len(sona)-(jadapikkus-1))]
    koikjadad+=sonajadad
print(koikjadad)

from collections import Counter
c=Counter(koikjadad)
print(c)
```

Tulemus

```
python3.5 paarid2.py

['all', 'llv', 'lve', 'vee', 'ees', 'esp', 'spo', 'por', 'ort', 'spo', 'por', 'ort', 'rtl',
'tla', 'lan', 'ane', 'mee', 'ees', 'esk', 'sko', 'kon', 'onn', 'nna', 'nas', 'asp', 'spo',
'por', 'ort', 'tug', 'ugi', 'git', 'ito', 'too', 'ool', 'oli', 'lis', 'isp', 'spo', 'por',
'ort']
Counter({'ort': 4, 'spo': 4, 'por': 4, 'ees': 2, 'all': 1, 'isp': 1, 'vee': 1, 'kon': 1,
'oli': 1, 'lve': 1, 'llv': 1, 'ugi': 1, 'mee': 1, 'ool': 1, 'tug': 1, 'ito': 1, 'ane': 1,
'nna': 1, 'lan': 1, 'lis': 1, 'too': 1, 'tla': 1, 'nas': 1, 'onn': 1, 'git': 1, 'esp': 1,
'rtl': 1, 'sko': 1, 'esk': 1, 'asp': 1})
```

Harjutus

- Pane näited käima
- Leia levinumad tähtede kolmikud sõnadest laulma, regilaul, laulupidu
- Leia levinumad tähtede paarid samadest sõnadest

Erinevuse üldistatavus

Järjendite või sõnade suhteline sagedus ikka mõnevõrra kõigub. Testi abil aga saab hinnata, kuivõrd erinevus on üldistatav. Üheks mooduseks on 2x2 hii-ruut-test. Pythoni `scipy`-paketi `stats` mooduli `chi2_contingency` käskluse puhul tuleb andmed sisse anda kujul, kus mõlema uuritava andmestiku kohta on kirjas vaatluse all olevate järjendite arv ning kõigi muude järjendite arv. Siin näites võrdleme kahes tekstis sõna "kui" esinemissagedust

```
          kui       muu
tekst1    3         5
tekst2    0         7
```

ehk siis paar `[kui1, mittekui1]` on `[3, 5]` ning `[kui2, mittekui2]` on `[0, 7]`

Käsk anna väljundina loetelu, mille element nr 1 on võrdluse p-väärtus, ehk tõenäosus, et mõõdetava üksuse (praegusel juhul sõna "kui") sageduse erinevus on juhuslik (nullhüpotees).

```
from scipy import stats
tekst1="nii sina kui tema kui meie kui teie".split(" ")
tekst2="juhan vana naljahammas pikad saapad jalga tombas".split(" ")
kui1=tekst1.count("kui")
kui2=tekst2.count("kui")
mittekui1=len(tekst1)-kui1
mittekui2=len(tekst2)-kui2
print(kui1, kui2, mittekui1, mittekui2)
print(stats.chi2_contingency([[kui1, mittekui1], [kui2, mittekui2]])[1])
```

Käivitus:

```
$ python3.5 paarid3.py
3 0 5 7
0.244227262957
```

Vastusest paistab, et sarnaselt "kui"-sid kasutavate tekstide puhul võib 24% tõenäosusega juhtuda selline erinevus sageduses. See annab küll vihje, et siin võib olla midagi erinevat, aga veerandiga eksimisvõimalus on liiga suur millegi kindlama väitmiseks. Tüüpiliselt piirdutakse 5%, vahel ka 1% eksimisvõimalusega. Muutes esimest teksti veelgi kui-rikkamaks, näidatakse juhusliku sattumise tulemuseks 2,77%, mis ütleb, et nõnda suur sageduste erinevus juba naljalt kogemata ei teki.

```
tekst1="nii sina kui tema kui meie kui teie kui kui kui kui kui".split(" ")
0.0277372888154
```

Kui tekstid on pikemad ja andmeid rohkem, siis muutub ka juba väiksema suhtelise erinevuse puhul uuritava järjendi erinevus üldistatavaks.

Kahe teksti tähepaaride võrdlemine

Sõnade kasutuse üldistatava erinevuse uurimist nimetatakse võtmesõnade leidmiseks - näeb, millised sõnad kus tekstis üldistatavalt sagedamini esile tulevad. Sarnaselt aga saab võrrelda tähepaare/kolmikuid, silpe, nootide järjestust, pöörangu- ja teivasjaamade asetumist ning pea kõike üksiku või järjestuslõikude kaupa loendatavat.

Alustuseks küsime välja tähepaarid Kungla rahva laulu sõnadest. Asendame kõik peale tähtede tühikutega ning siis tükeldame teksti tühikute kohalt.

```
import urllib.request
import re
jadapikkus=2
def jadad(aadress):
  tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()
  tekst=re.sub(r'[^a-zõäöü]', ' ', tekst)
  print(tekst)
```

```
  sonad=tekst.split(' ')
  koikjadad=[]
  for sona in sonad:
    sonajadad=[sona[arv:arv+jadapikkus] for arv in range(len(sona)-(jadapikkus-1))]
    koikjadad+=sonajadad
  return koikjadad

jadad1=jadad("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt")
print(jadad1)
```

Väljundiks tekst

```
kui kungla rahvas kuldsel aal  kord istus maha sööma   siis vanemuine murumaal  läks kandle
lugu lööma     läks aga metsa mängima   läks aga laande lauluga     sealt saivad lind ja
lehepuu  ja loomad laululugu   siis laulis mets ja mere suu  ja eesti rahva sugu     siis
kõlas kaunis lauluviis  ja pärjad pandi pähe   ja murueide tütreid siis  sai eesti rahvas
näha     ma laulan mättal  mäe peal  ja õhtu hilja õues  ja vanemuise kandle hääl   see
põksub minu põues
```

ja paarid

```
['ku', 'ui', 'ku', 'un', 'ng', 'gl', 'la', 'ra', 'ah', 'hv', 'va', 'as', 'ku', 'ul', 'ld',
'ds', 'se', 'el', 'aa', 'al', 'ko', 'or', 'rd', 'is', 'st', 'tu', 'us', 'ma', 'ah', 'ha',
'sö', 'öö', 'öm', 'ma', 'si', 'ii', 'is', 'va', 'an', 'ne', 'em', 'mu', 'ui', 'in', 'ne',
'mu', 'ur', 'ru', 'um', 'ma', 'aa', 'al', 'lä', 'äk', 'ks', 'ka', 'an', 'nd', 'dl', 'le',
'lu', 'ug', 'gu', 'lö', 'öö', 'öm', 'ma', 'lä', 'äk', 'ks', 'ag', 'ga', 'me', 'et', 'ts',
'sa', 'mä', 'än', 'ng', 'gi', 'im', 'ma', 'lä', 'äk', 'ks', 'ag', 'ga', 'la', 'aa', 'an',
'nd', 'de', 'la', 'au', 'ul', 'lu', 'ug', 'ga', 'se', 'ea', 'al', 'lt', 'sa', 'ai', 'iv',
'va', 'ad', 'li', 'in', 'nd', 'ja', 'le', 'eh', 'he', 'ep', 'pu', 'uu', 'ja', 'lo', 'oo',
'om', 'ma', 'ad', 'la', 'au', 'ul', 'lu', 'ul', 'lu', 'ug', 'gu', 'si', 'ii', 'is', 'la',
'au', 'ul', 'li', 'is', 'me', 'et', 'ts', 'ja', 'me', 'er', 're', 'su', 'uu', 'ja', 'ee',
'es', 'st', 'ti', 'ra', 'ah', 'hv', 'va', 'su', 'ug', 'gu', 'si', 'ii', 'is', 'kõ', 'õl',
'la', 'as', 'ka', 'au', 'un', 'ni', 'is', 'la', 'au', 'ul', 'lu', 'uv', 'vi', 'ii', 'is',
'ja', 'pä', 'är', 'rj', 'ja', 'ad', 'pa', 'an', 'nd', 'di', 'pä', 'äh', 'he', 'ja', 'mu',
'ur', 'ru', 'ue', 'ei', 'id', 'de', 'tü', 'üt', 'tr', 're', 'ei', 'id', 'si', 'ii', 'is',
'sa', 'ai', 'ee', 'es', 'st', 'ti', 'ra', 'ah', 'hv', 'va', 'as', 'nä', 'äh', 'ha', 'ma',
'la', 'au', 'ul', 'la', 'an', 'mä', 'ät', 'tt', 'ta', 'al', 'mä', 'äe', 'pe', 'ea', 'al',
'ja', 'õh', 'ht', 'tu', 'hi', 'il', 'lj', 'ja', 'õu', 'ue', 'es', 'ja', 'va', 'an', 'ne',
'em', 'mu', 'ui', 'is', 'se', 'ka', 'an', 'nd', 'dl', 'le', 'hä', 'ää', 'äl', 'se', 'ee',
'põ', 'õk', 'ks', 'su', 'ub', 'mi', 'in', 'nu', 'põ', 'õu', 'ue', 'es']
```

Eristuvate üksuste leidmiseks tuleb üht andmestikku võrrelda teisega. Siinsel juhul võtame Kungla rahva loo võrdluseks naljajutu lambipirni kohta. Sama alamprogramm aitab tekstist paarid leida, edasi tuleb iga paari kohta kokku lugeda, mitu seda kummaski tekstis leidub ning hii-ruudu valemi järgi vaadata, kas erinevus on üldistatav.

```
import urllib.request
import re
from collections import Counter
from scipy import stats
jadapikkus=2
def jadad(aadress):
  tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()
  tekst=re.sub(r'[^a-zõäöü]', ' ', tekst)
  print(tekst)
  sonad=tekst.split(' ')
```

```
  koikjadad=[]
  for sona in sonad:
    sonajadad=[sona[arv:arv+jadapikkus] for arv in range(len(sona)-(jadapikkus-1))]
    koikjadad+=sonajadad
  return koikjadad

jadad1=jadad("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt")
jadad2=jadad("http://www.tlu.ee/~jaagup/andmed/keel/lambipirn.txt")
kogused1=Counter(jadad1)
kogused2=Counter(jadad2)
for jada in kogused1.keys():
  jada1=kogused1[jada]
  mittejada1=len(jadad1)-jada1
  jada2=0
  mittejada2=len(jadad2)
  if jada in kogused2:
    jada2=kogused2[jada]
    mittejada2=len(jadad2)-jada2
  print(stats.chi2_contingency([[jada1, mittejada1], [jada2, mittejada2]])[1], jada)
```

Näites võrdlen esialgu vaid Kungla rahva juures esinenud paare teise tekstiga. Nagu algusotsast näha, siis `au` on hästi eristuv, `lä` ka, samas `mi` ning `so` esinevad mõlemis tekstis võrreldavalt ning mõõdetavat erisust ei teki.

```
0.0449309837046 lä
0.832911553834 mi
0.420355516352 sö
0.00112842065061 au
…
```

Mõnevõrra täiendatud koodi teine pool: tsüklis vaatan ühisosa mõlema teksti tähepaaridest. Arvutan paaride osakaalu kummaski tekstis ning hiljem järjestan osakaalude vahe järgi. Nii on võimalik jälgida, kas neid on Kungla rahva loos rohkem või vähem. Kõigi massiivi elementide välja trükkimisel `print(*r)` abil asendab tärn listi `r` üksikud liikmed print käsu parameetriteks. Kirjapilt tähendab sama, kui `print(r[0], r[1], r[2], r[3])`

```
jadad1=jadad("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt")
jadad2=jadad("http://www.tlu.ee/~jaagup/andmed/keel/lambipirn.txt")
kogused1=Counter(jadad1)
kogused2=Counter(jadad2)
hoidla=[]
for jada in set(kogused1.keys()).union(set(kogused2.keys())):
  jada1=0
  mittejada1=len(jadad1)
  jada2=0
  mittejada2=len(jadad2)
  if jada in kogused1:
    jada1=kogused1[jada]
    mittejada1=len(jadad1)-jada1
  if jada in kogused2:
    jada2=kogused2[jada]
    mittejada2=len(jadad2)-jada2
  sarnasustoenaosus=stats.chi2_contingency([[jada1, mittejada1], [jada2, mittejada2]])[1]
  if sarnasustoenaosus<=0.1:
    osakaal1=round(jada1/len(jadad1), 3)
    osakaal2=round(jada2/len(jadad2), 3)
```

```
    suund="+" if osakaal1>osakaal2 else "-"
    hoidla.append([jada, suund, osakaal1, osakaal2, round(sarnasustoenaosus, 3)])

for r in sorted(hoidla, key=lambda rida: rida[2]-rida[3]):
  print(*r)
```

Silp te paistab olema loetelus ainus, mis lambipirni jutus suurema sagedusega silma paistab - Kungla rahvas see puudub ning Lambipirni loos on sagedusega 1,5%. Tõenäosusega 7,4% on erinevus juhuslik, teistpidi siis vähemasti 90% kindlusega võime väita, et erinevus näitab lambipirni teksti eripära.

```
te - 0.0 0.015 0.074
öö + 0.007 0.001 0.043
dl + 0.007 0.0 0.001
öm + 0.007 0.0 0.001
ha + 0.007 0.0 0.001
me + 0.011 0.002 0.071
lä + 0.011 0.002 0.045
ui + 0.011 0.002 0.071
äk + 0.011 0.001 0.001
mä + 0.011 0.001 0.005
ug + 0.014 0.003 0.021
ue + 0.011 0.0 0.0
hv + 0.011 0.0 0.0
mu + 0.014 0.003 0.021
nd + 0.018 0.006 0.054
ii + 0.018 0.005 0.022
lu + 0.018 0.002 0.0
au + 0.021 0.004 0.001
an + 0.025 0.007 0.004
ja + 0.035 0.016 0.031
is + 0.032 0.012 0.017
la + 0.032 0.011 0.009
ul + 0.025 0.004 0.0
```

Harjutus

- Leia regiviiside tabelist kolm esimest nooti lugudel, mille tugiheli on g http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt
- Leia regiviiside tabelist kolm esimest nooti lugudel, mille tugiheli on c
- Leia alguskolmikute sagedused kummagi helistiku juures
- Leia Hii-ruut võrdluse abil, millised on kummagi helistiku jaoks eristuvad alguskolmikud

- Võrdle kummagi tugiheliga lugude juures libisevalt nelja nooti - leia eristavad järgnevused

Alguskolmikud

```
import urllib.request
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
```

```
   ).read().decode("utf8").split("\n")[1:]
gviisikolmikud=[rida.split(',')[10:13] for rida in viisid if rida.split(',')[6]=='g']
print(gviisikolmikud[:5])

[['2d', 'h', 'h'], ['2d', '2d', '2c'], ['2d', '2d', '2d'], ['g', 'd', '2c'], ['h', '2d',
'h']]
```

Alguskolmikud ja nende sagedused g ja c-duuris. `Counter`i objekt lubab levinumad sagedused välja kuvada `most_common` käsuga. Andmestikust vastava jada sageduste ja ülejäänud elementide sageduste välja näitamiseks lisati funktsioon `jahei` - nii ei pea sama lõiku mitmel korral kirjutama.

```
import urllib.request
from collections import Counter
from scipy import stats
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
   ).read().decode("utf8").split("\n")[1:]
gviisikolmikud=["-".join(rida.split(',')[10:13])
                for rida in viisid if rida.split(',')[6]=='g']
gkogused=Counter(gviisikolmikud)
print(gkogused.most_common(5))

cviisikolmikud=["-".join(rida.split(',')[10:13])
                for rida in viisid if rida.split(',')[6]=='c']
ckogused=Counter(cviisikolmikud)
print(ckogused.most_common(5))

def jahei(jada, kogused, kolmikud):
  jah=kogused[jada] if jada in kogused else 0
  ei=len(kolmikud)-jah
  return [jah, ei]

for jada in set(gkogused+ckogused):
  jeg=jahei(jada, gkogused, gviisikolmikud)
  jec=jahei(jada, ckogused, cviisikolmikud)
  p=stats.chi2_contingency([jeg, jec])[1]
  if p<0.01: print(jada, p, jeg, jec)
```

Eri helistike viiside algusnoodid eristuvad hästi - nii kuvame välja vaid need algused, mis vähemasti 99% tõenäosusega on vastavale helistikule eripärased võrreldes paariliseks valitud helistikule. Nagu aga näha, siis g-g-a on aga mõlemi helistiku puhul jällegi suhteliselt levinud algus - seega eristavate alla ei kuulu.

```
 $ python3.5 viisijupid2.py
[('g-g-g', 149), ('g-a-h', 92), ('g-g-a', 87), ('2d-2d-2c', 83), ('2d-h-2d', 80)]
[('c-e-g', 35), ('2c-h-2c', 27), ('g-g-a', 20)]
g-h-2d 8.02231221191e-05 [77, 2169] [1, 536]
g-e-c 6.07235470694e-05 [0, 2246] [5, 532]
c-e-c 6.88961171888e-06 [0, 2246] [6, 531]
2d-h-h 0.000119138698246 [74, 2172] [1, 536]
g-a-h 0.000223128304399 [92, 2154] [4, 533]
2c-2e-2c 2.61656797782e-10 [4, 2242] [15, 522]
2e-2d-2c 1.18993385422e-07 [3, 2243] [11, 526]
```

Libisevalt leitud lõigud

Viimatises näites piirduti viiside algustega. Teistmoodi ülevaate viisi liikumisest saab, kui võtta ette viisist lõigud "akendena". Neljase akna puhul noodid 1-4, 2-5, 3-6 jne kuni lõpuni välja, 13-16. Nii tulevad kõrvuti olevad järjestused välja sõltumata nende asukohast viisis.

```
andmed={"kogus":0}
f=open("regiviisid.txt")
f.readline()
jadapikkus=4
viisipikkus=16
for rida in f:
 if rida.strip().split(",")[6]=='g':
  viis=rida.strip().split(",")[10:10+viisipikkus]
  jadad=[viis[arv:arv+jadapikkus] for arv in range(viisipikkus-(jadapikkus-1))]
  for jada in jadad:
   plokk=andmed
   plokk["kogus"]+=1
   for noot in jada:
    if noot in plokk:
     plokk[noot]["kogus"]+=1
    else:
     plokk[noot]={"kogus":1}
    plokk=plokk[noot]
f.close()

minkogus=100
f2=open("vastus.txt", "w")
#print(".\r", file=f2)
def kuva(plokk, taane):
   for kogus, noot in sorted([[plokk[noot]["kogus"], noot] \
                              for noot in plokk if not noot=="kogus"], reverse=True):
    if kogus>=minkogus:
      print(taane*"\t"+noot+"_"+str(kogus)+"\r", file=f2)
      kuva(plokk[noot], taane+1)

kuva(andmed, 0)
f2.close()
```

Väljundi algusots

```
a_6445
	a_1915
		g_689
			g_476
			h_151
		a_459
			a_129
		f_180
		2c_176
		h_174
	g_1545
		g_980
			g_824
		h_218
		a_125
	h_1068
```

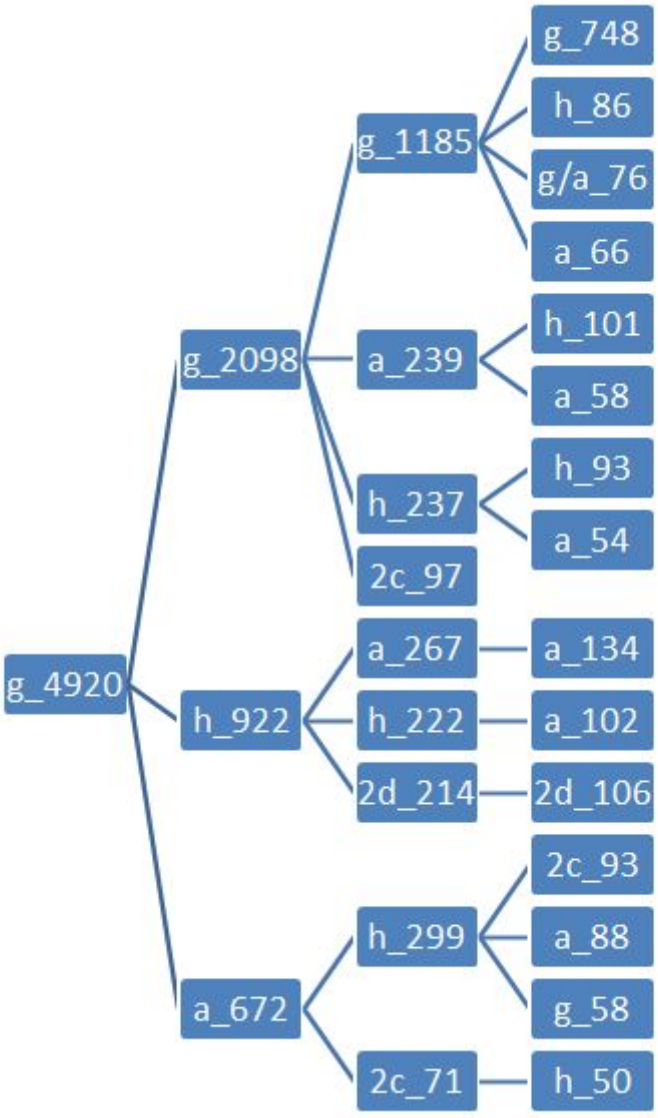
```
                g_309
                        g_189
                a_281
                        g_111
                2c_220
                h_126
        2c_641
                h_254
                a_177
                2c_129
        f_504
                a_209
                g_113
        2d_262
h_6213
        a_1957
                a_670
                        g_391
                        a_128
```

Väljund tervikuna näha aadressil

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/noodipuu1.txt`

Nagu paistab, siis g-duuri viiside juures on levinuimad hoopis a-noodi ehk teise astmega algavad neljalised lõigud, neile järgnevad h-noodi ehk kolmanda astmega algavad lõigud ning alles kolmandana tulevad g-noodi enesega algavad lõigud.

G-ga algavatest lõikudest tehti aga suuremas kirjas illustreeriv puu, kus levinumad neljanoodilised järjestused näha. Ehkki need algavad viisist suvalistel kohtadel, siis mõttes läbi ümisedes tunduvad kõik meeldivad ja loogilised olema.



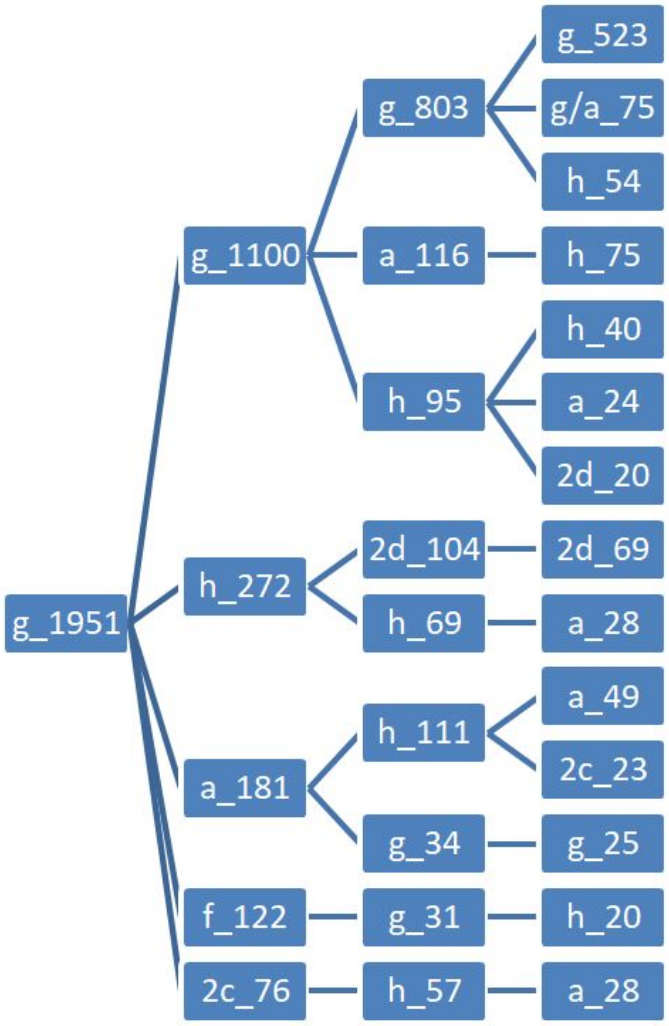
Rõhutatud kohtadest algavad lõigud

Eelmises näites võeti jadadeks kõik sobiva pikkusega lõigud ning viisilõigud tundusid juba seal meeldivad olema. Andmestikus olevad kaherealised kuueteistkümnenoodilised viisid jagunevad aga kõigepealt kaheks kaheksanoodiliseks reaks ning edasi sageli kumbki rida kaheks omaette rõhutatud pooleks. Nii võtame siin näites jadade alguseks iga neljanda noodi, ehk noodid nullist lugedes järjekorranumbritega 0, 4, 9 ja 12

```
jadad=[viis[arv:arv+jadapikkus] for arv in [0, 4, 9, 12]]
```

Viisilõike eelnevaga võrreldes märgatavalt vähem, samas need nüüd enamikus rõhulise algusega kohtadest võetud ning võiksid seetõttu terviklikuma neljanoodilise lõigu välja anda. Väljundit puhastades jäeti praegu sisse vaid harud, mida nelja noodi ulatuses leidus

vähemalt kahekümnel korral - nii võimalik neid oma kõrvadele terviklikumalt ja järjest läbi laulda.



Harjutus

- Tee näited läbi
- Tee samad näited läbi C-duuri põhiheliga viisidega. Näita, milline andmepuu tekib viisialguste puhul, milline libisevate jadade ning milline 1., 4., 9. ja 12. löögil algavate lõikude puhul

Graaf ehk võrgustik

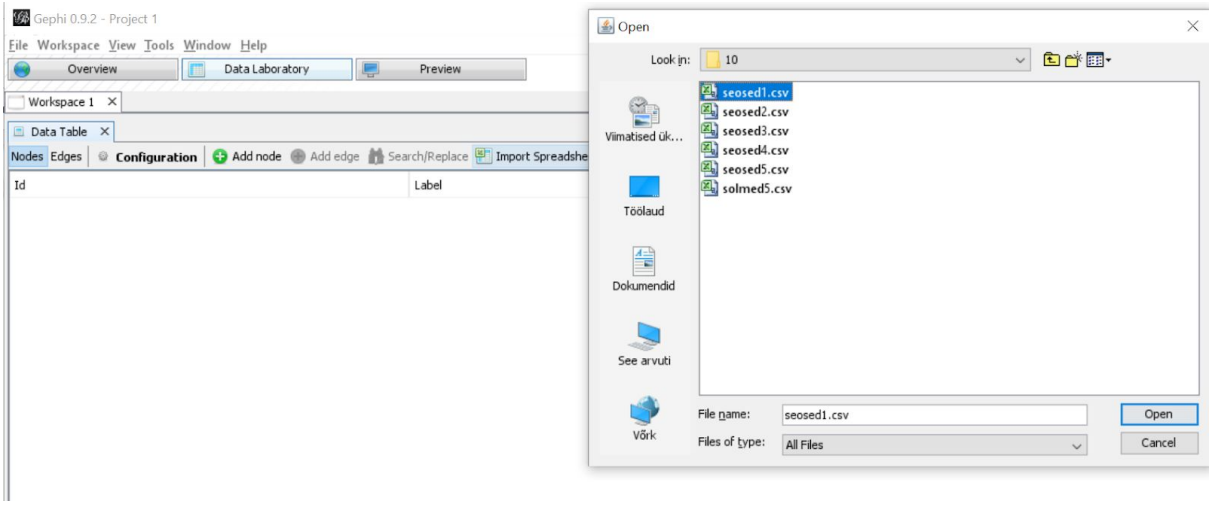
Puu näitab hästi hargnevaid andmejadasid. Kui punktid on läbisegi üksteisega seotud, siis üheks võimaluseks seoseid näha on graaf.

Gephi

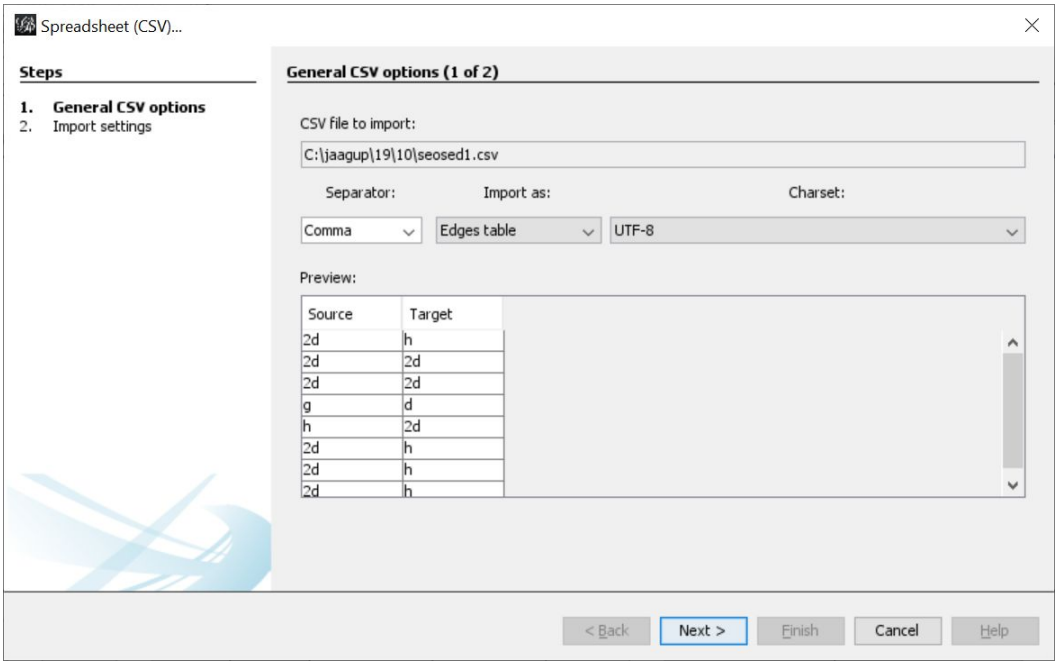
Üheks levinud graafiandmetega tutvumise rakenduseks on vabalt installitav rakendus Gephi (gephi.org). Näidisena paneme sinna sisse kaheksa regilauluviisi kaks esimest nooti. Tulpade pealkirjadeks Gephi soovide järgi Source ning Target - ehk siis kust ja kuhu viis liigub. Kui andmeid vaadata, siis viis paari on 2d ja h vahel, kaks viisi algavad kahe 2d-ga ning üks seob omavahel g ja d.

```
Source,Target
2d,h
2d,2d
2d,2d
g,d
h,2d
2d,h
2d,h
2d,h
```

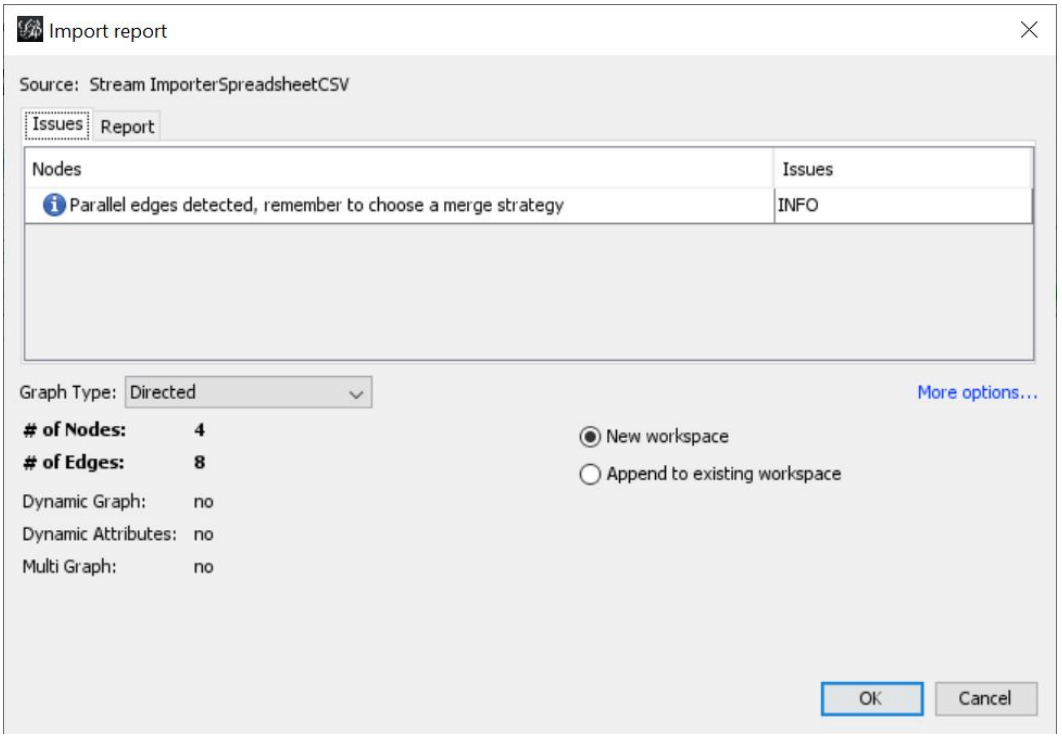
Andmete rakenduse sisse saamiseks tuleb fail salvestada csv-laiendiga - näiteks seosed1.csv. Luua Gephis uus projekt, valida sealt Data Laboratory -> Edges ning Import Spreadsheet



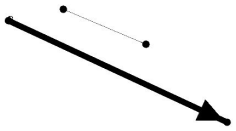
Näidatakse, millised tulbad failist leiti.



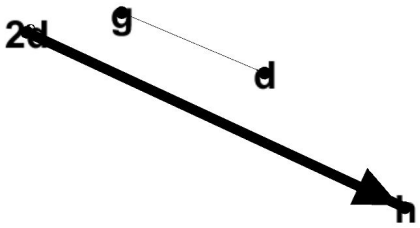
Lõpus küsitakse üle, kuhu saadud andmed paigutada. Esialgu võib valida "New workspace". Kui hiljem tahetakse samasse projekti ja töölehele andmeid juurde tuua, siis tuleb võtta alumine valik olemasolevatele andmetele lisamiseks.



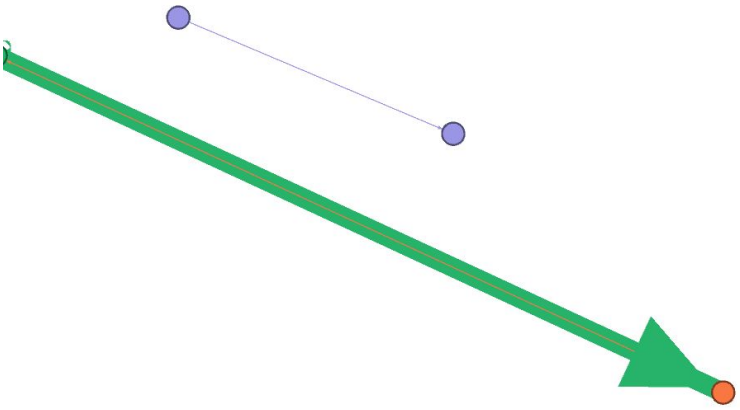
Vajutades `Overview` peale, kuvatakse tekkinud joonis. Võib aimata, et jämedamad jooned on tugevamate seoste vahel.



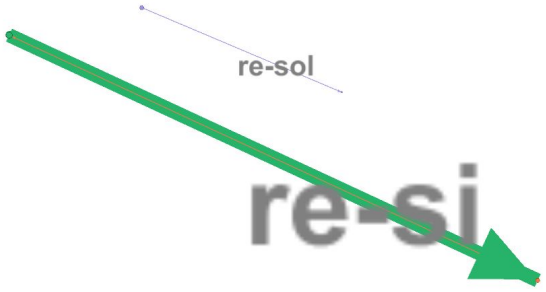
Noodinimede nägemiseks tuleb Nodes sakilt valida "`Copy data to other column`" ning kopeerida `Id`-tulba andmed `Label`-tulpa. Siis saab `Overview` alt T-tähe juurest tekstid sisse lülitada ninga paistab lähemalt, et millega tegu.



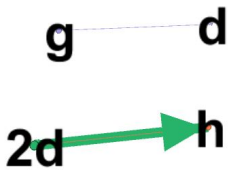
Hiirega rullikuga kerides võimalik pilti lähemalt uudistada, samuti vastavalt soovidele värvida.



Kui seostele (`edges`) kirjutada tabelisse Label-ossa väärtused, siis saab needki joonistel kuvama panna.



Mõnevõrra sättides saab tulemuse silmale paremini loetavamaks.



Harjutus

- Tee näide läbi
- Leia tähepaarid sõnas kalamaja, paiguta csv-faili. Näita Gephi abil seosed tähtede vahel

Seoste faili genereerimine

Algusotsa proovisime käsitsi. Kogu regilaulude faili läbi käimiseks on aga koodilõik sõbralik abiline. Kuvatakse kõik G-duuri viiside kahest esimesest noodist tekkivad seosed

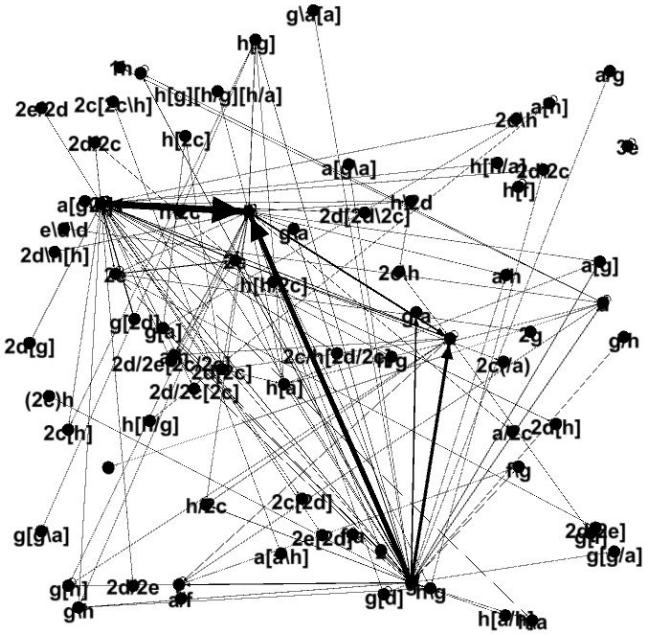
```
import urllib.request
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
  ).read().decode('utf8').split("\n")[1:]
f=open("seosed2.csv", "w")
print("Source,Target", file=f)
for viis in viisid:
```

```
  m=viis.split(",")
  if m[6]=="g":
    print(m[10]+","+m[11], file=f)
f.close()
```

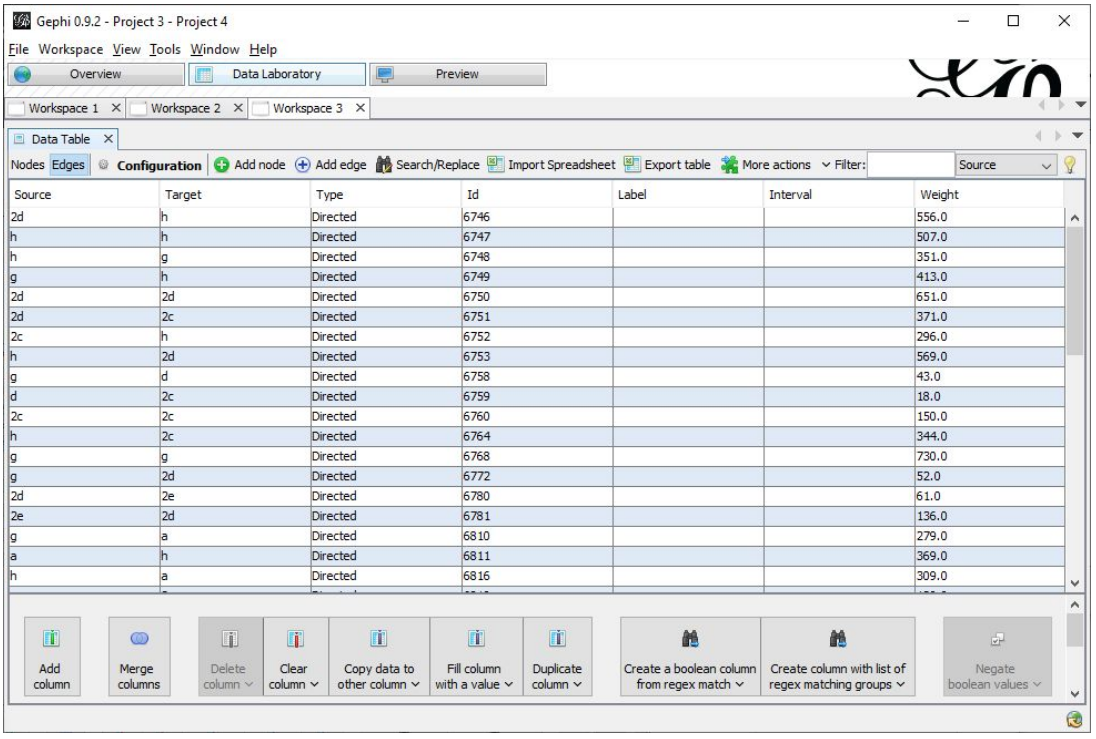
Tulemus

```
Source,Target
2d,h
2d,2d
2d,2d
g,d
h,2d
2d,h
2d,h
2d,h
2d,h
h,2d
h,2d
h,2d
```

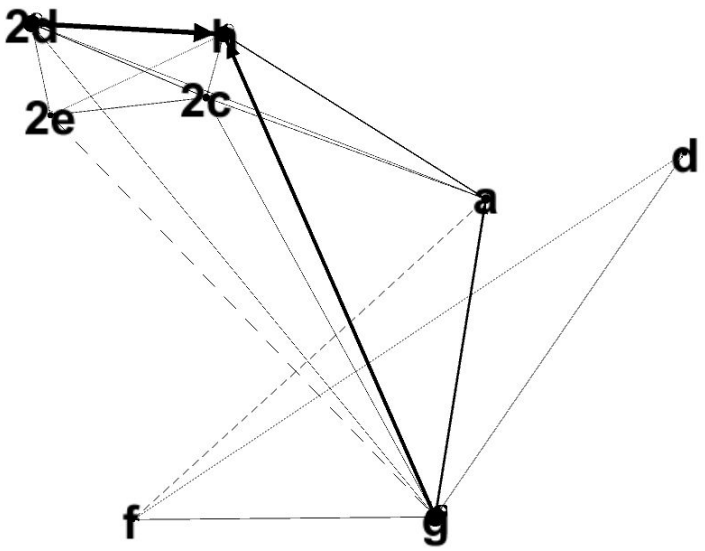
Esimese hooga sisse lugedes tuleb pilt mõnevõrra kirju, siiski näha mõned suuremad nooled levinumate seoste kohta



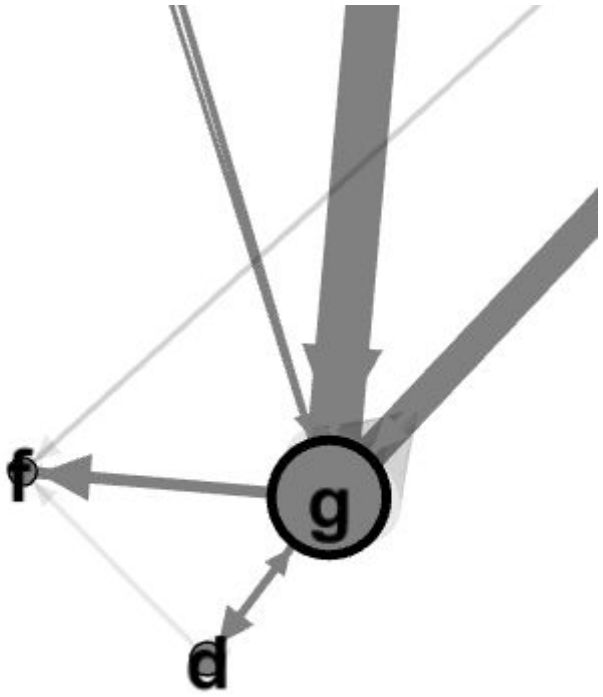
`Data Laboratory` alt jätame alles vaid seosed, mida on rohkem - nii saab ka pildi selgemaks. Seosed saab kaalu (ehk praegusel juhul arvu) järgi järjestada ning siis alumised maha kustutada.



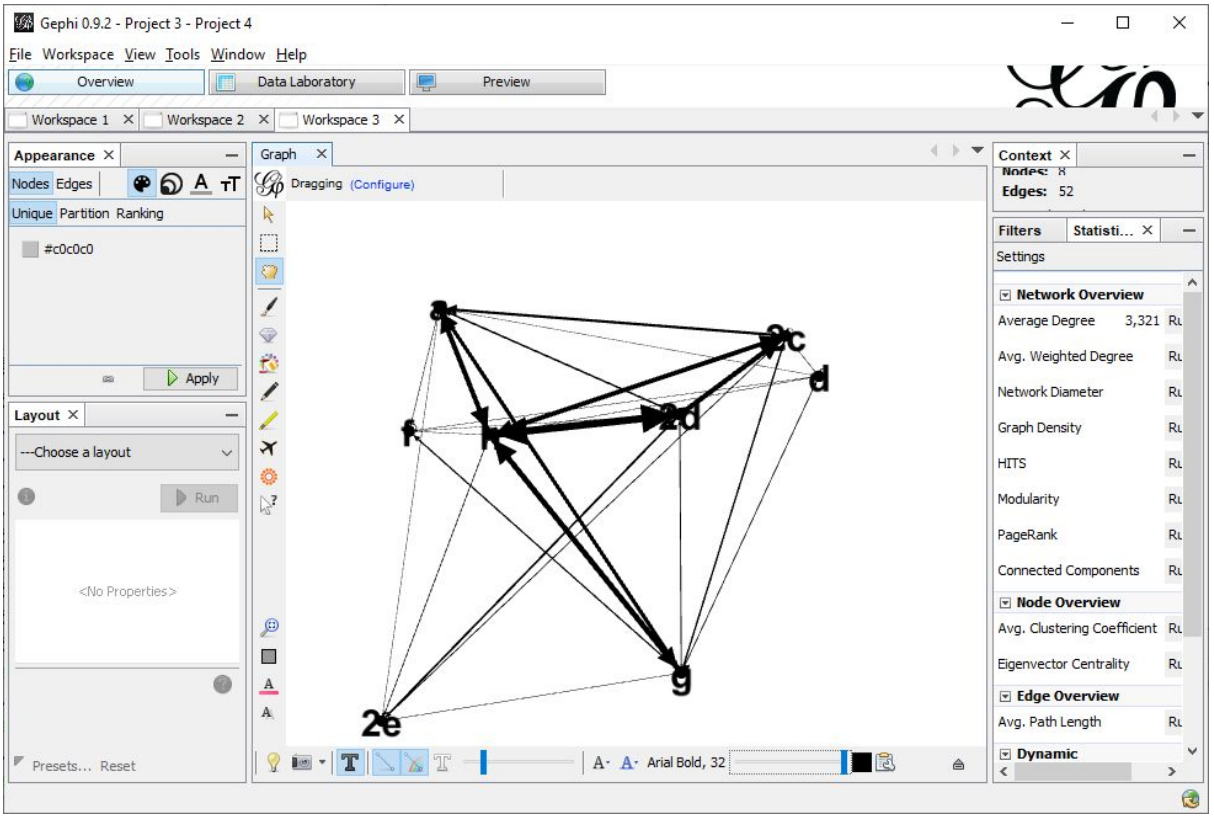
Pilt saab selgemaks



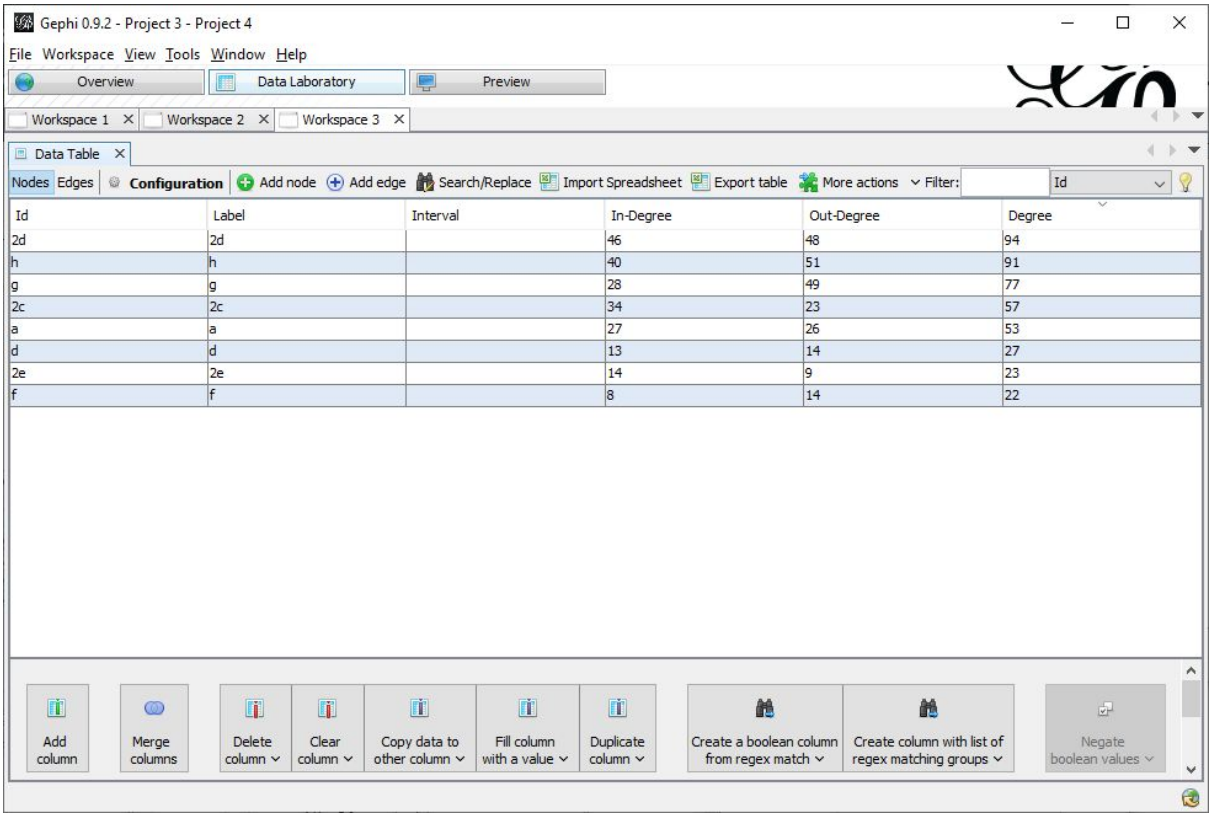
Lähemalt välja suurendades paistab, kuidas seosed G-noodi juurest liiguvad.



Enamkasutatavate sõlmede jaoks saab paremalt välja arvutada "`Average Degree`" - ehk kui palju ühendusi millise sõlme juures on.



Tulemusena tekib iga sõlme juurde tulp "`Degree`", mille järgi leiab üles levinumad sõlmed ja võib nad alles jätta.



Harjutus

- Tee näide läbi
- Koosta programm, mis salvestab etteantud tekstist tähepaarid faili. Koosta nendest Gephi abil joonis

Viisilõik graafina

Võrgustik võimaldab näidata, et kui palju ühest kõrguselt teise üleminekuid millise noodi juures on. Selleks valime g-duuri viisid ning märgime esimese nelja noodi juures noodipaarid. Sõlmed tähistame koos löögi järjekorranumbriga. Paarid 2d_1,h_2 ning h_2,h_3 tähendavad, et viisi esimene noot oli 2d, teine h ning kolmas ka h - kokku moodustavad nad kaks järjestikust paari.

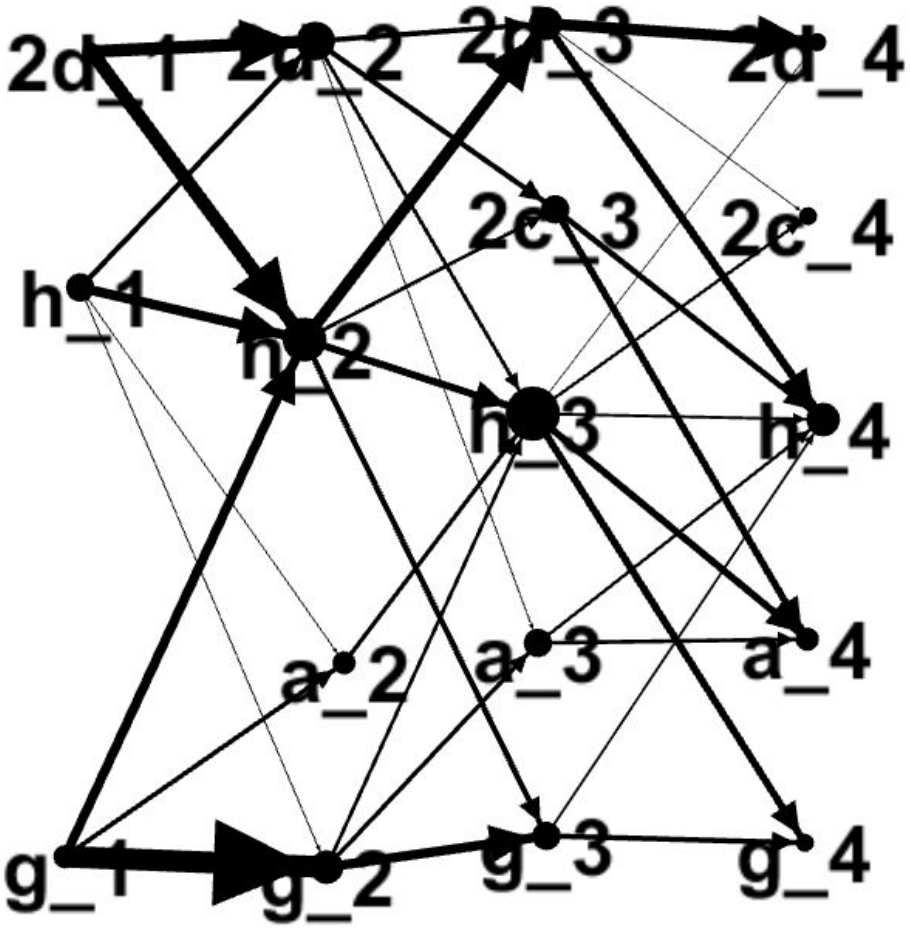
```
import urllib.request
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
  ).read().decode('utf8').split("\n")[1:]
f=open("seosed4.csv", "w")
print("Source,Target", file=f)
for viis in viisid:
  m=viis.split(",")
```

```
  if m[6]=="g":
    for algus in range(10, 14):
      print(m[algus]+"_"+str(algus-9)+","+m[algus+1]+"_"+str(algus-8), file=f)
f.close()
```

Tulemus:

```
Source,Target
2d_1,h_2
h_2,h_3
h_3,g_4
g_4,h_5
2d_1,2d_2
2d_2,2c_3
2c_3,h_4
h_4,2d_5
2d_1,2d_2
2d_2,2d_3
2d_3,2c_4
2c_4,h_5
g_1,d_2
d_2,2c_3
```

Kui graaf Gephi abil välja joonistada, levinumad sõlmed alles jätta ning sõlmed löökide järgi paika tõsta, saab ligikaudu sarnase tulemuse:



Nagu näha, on loo algul levinud noodipaarid g-g ning 2d-h. Edasi saab samuti vaadata, et kui levinult viisid liiguvad.

Algus ja ots

Lõigu alguse ja lõpu selgemaks kuvamiseks ning sõlmede mugavamaks tõstmiseks on vahel hea luua eraldi "olematud" sõlmed alguse ja otsa tarbeks. Et ei peaks käsitsi haruldasemaid sõlmi eraldama, jätame koodi vaid kontrolli, millise nimega sõlmedest seosed alles jätta.

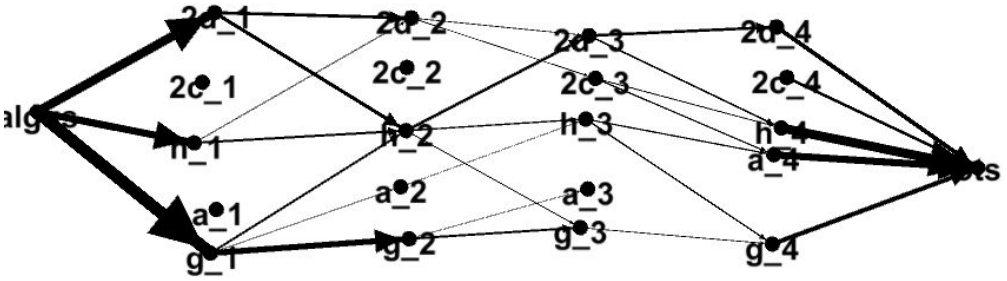
```
import urllib.request
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
  ).read().decode('utf8').split("\n")[1:]
alles={"g", "a", "h", "2c", "2d", "algus", "ots"}
f=open("seosed5.csv", "w")
print("Source,Target", file=f)
for viis in viisid:
  m=viis.split(",")
  if m[6]=="g":
    for algus in range(10, 13):
     if m[algus] in alles and m[algus+1] in alles:
      if algus==10: print("algus,"+m[algus]+"_1", file=f)
      print(m[algus]+"_"+str(algus-9)+","+m[algus+1]+"_"+str(algus-8), file=f)
```

```
      if algus==12: print(m[algus+1]+"_4,ots", file=f)
f.close()
```

Tulemus

```
Source,Target
algus,2d_1
2d_1,h_2
h_2,h_3
h_3,g_4
g_4,ots
algus,2d_1
2d_1,2d_2
2d_2,2c_3
2c_3,h_4
h_4,ots
algus,2d_1
2d_1,2d_2
2d_2,2d_3
2d_3,2c_4
2c_4,ots
```

Kuvamise ajal on võimalik algus ja ots soovitud paikadesse tõsta ning siis kohtadele kinnitada. Selleks tuleb sõlme peal valida hiire parema klahviga "Settle". Edasi näiteks Force Atlas 2 paigutus tõstab ülejäänud sõlmed enamvähem sobilikultpaika, mõnevõrra saab neid veel hiirega järele sättida.



Harjutus

- Tee näide läbi
- Leia mõned a-tähega algavad neljatähelised sõnad. Koosta nendest programmi abil sarnane sisendfail ja graaf

Koordinaatide järgi paigutus

Alles jäävatele sõlmedele määrame juba koodi sees y-koordinaadi. Nii need pärast hea samale kõrgusele sättida. Koostame nii seoste kui sõlmede faili.

```
import urllib.request
```

```
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
  ).read().decode('utf8').split("\n")[1:]
alles={"g":0, "a":10, "h":20, "2c":30, "2d":40}
f=open("seosed5.csv", "w")
print("Source,Target", file=f)
for viis in viisid:
  m=viis.split(",")
  if m[6]=="g":
    for algus in range(10, 13):
     if m[algus] in alles and m[algus+1] in alles:
      if algus==10: print("algus,"+m[algus]+"_1", file=f)
      print(m[algus]+"_"+str(algus-9)+","+m[algus+1]+"_"+str(algus-8), file=f)
      if algus==12: print(m[algus+1]+"_4,ots", file=f)
f.close()

f2=open("solmed5.csv", "w")
print("Id,Label,x,y", file=f2)
print("algus,algus,0,20", file=f2)
for nr in range(1, 5):
  for noot in alles:
     print(noot+"_"+str(nr)+","+noot+","+str(nr*10)+","+str(alles[noot]),file=f2)
print("ots,ots,50,20", file=f2)
f2.close()
```

Tulemus:

seosed5.csv

```
Source,Target
algus,2d_1
2d_1,h_2
h_2,h_3
h_3,g_4
g_4,ots
algus,2d_1
2d_1,2d_2
2d_2,2c_3
2c_3,h_4
h_4,ots
algus,2d_1
2d_1,2d_2
2d_2,2d_3
2d_3,2c_4
2c_4,ots
2c_3,2c_4
2c_4,ots
....
```

solmed5.csv

```
Id,Label,x,y
algus,algus,0,20
2c_1,2c,10,30
h_1,h,10,20
a_1,a,10,10
g_1,g,10,0
2d_1,2d,10,40
2c_2,2c,20,30
h_2,h,20,20
a_2,a,20,10
g_2,g,20,0
2d_2,2d,20,40
2c_3,2c,30,30
```

```
h_3,h,30,20
a_3,a,30,10
g_3,g,30,0
2d_3,2d,30,40
2c_4,2c,40,30
h_4,h,40,20
a_4,a,40,10
g_4,g,40,0
2d_4,2d,40,40
ots,ots,50,20
```

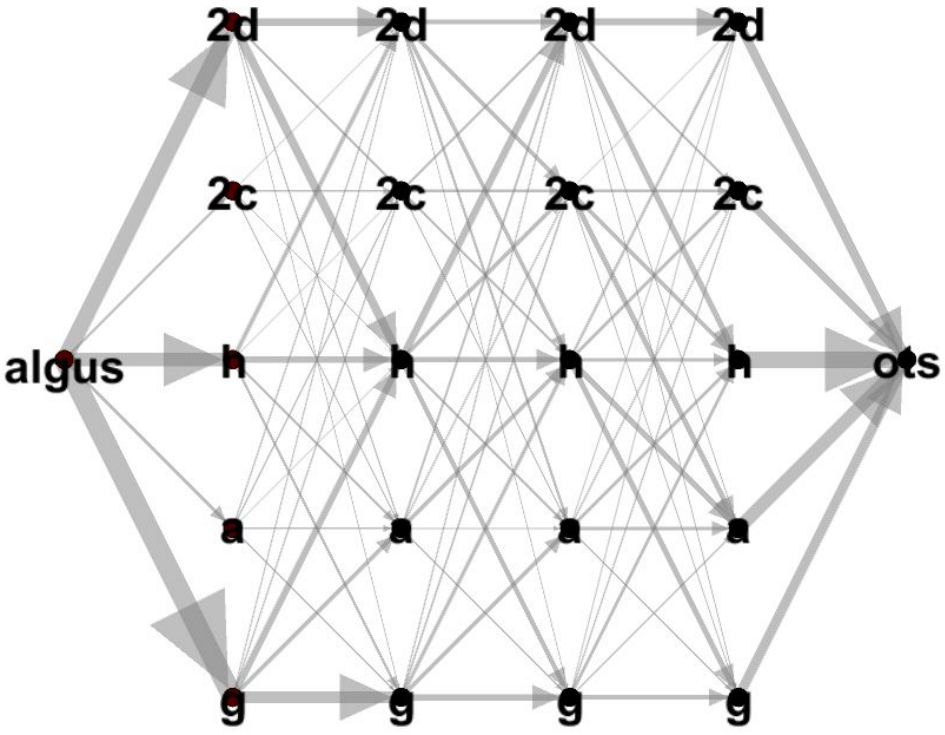
GeoLayouti installimine

Gephil on mitukümmend lisamoodulit. Neid lisada saab Tools-menüü Plugins käsu abil. Available loetelust leiab Geo Layout'i, mille siis paigaldada saab.

Tools->Plugins->Available-> Geo Layout - install

Layout -> Geo Layout. Latitude - y, Longitude - x

Võrreldese eelmise joonisega näeb joonis nüüd süstemaatilisem välja



Harjutus

- Koostage tekstifail sarnase kõlaga neljatähelistest sõnadest
- Koostage sõna tähtedest eelnenud viisinäitega sarnane graaf, kus näha, millistel kohtadel on rohkem üleminekuid ühest tähest teise.

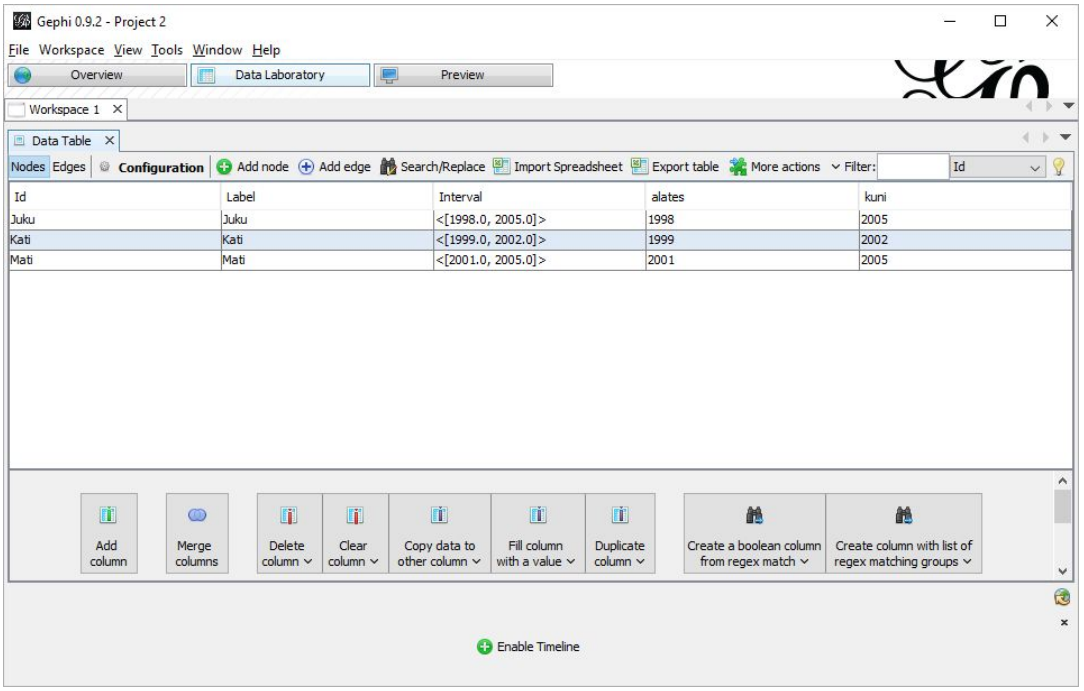
Animatsioon graafiga

Gephi võimaldab panna ühe parameetri (näiteks aja) muutuma ning siis kuvada jooniseid vastavalt sellele muutumisele. Isikute failis märgime kolme isiku tegutsemise ajad. Nagu näha, siis Juku ajavahemik on kõige pikem. Temaga koos asus tegutsema kõigepealt Kati. Mõne aja pärast lisandus Mati ning Kati lahkus.

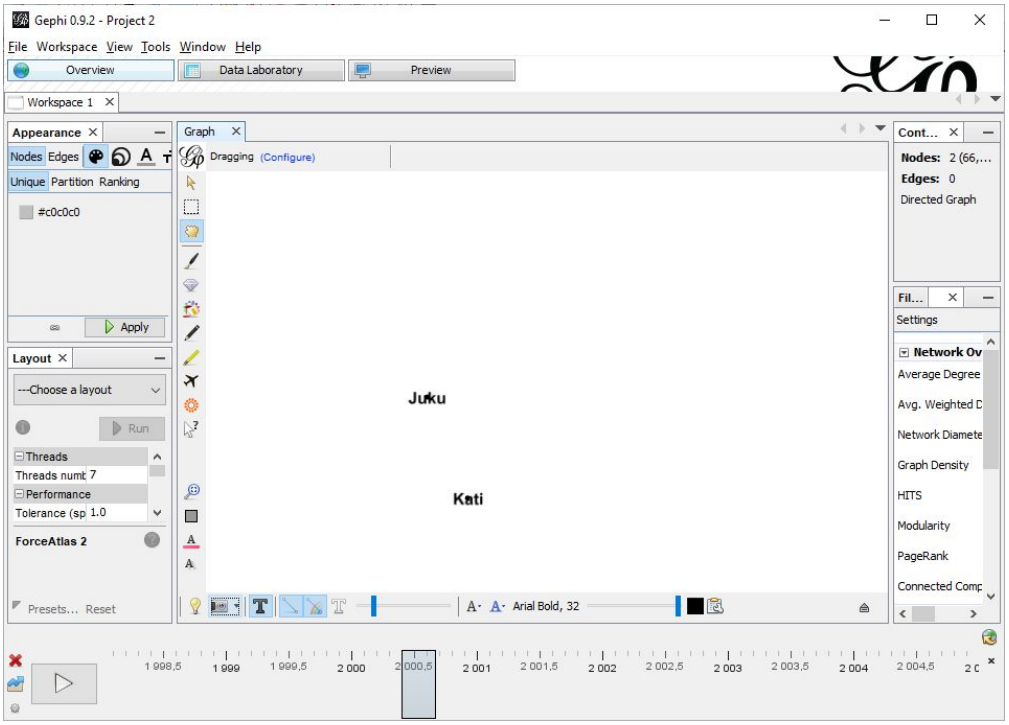
isikud.csv

```
Id,Label,alates,kuni
Juku,Juku,1998,2005
Kati,Kati,1999,2002
Mati,Mati,2001,2005
```

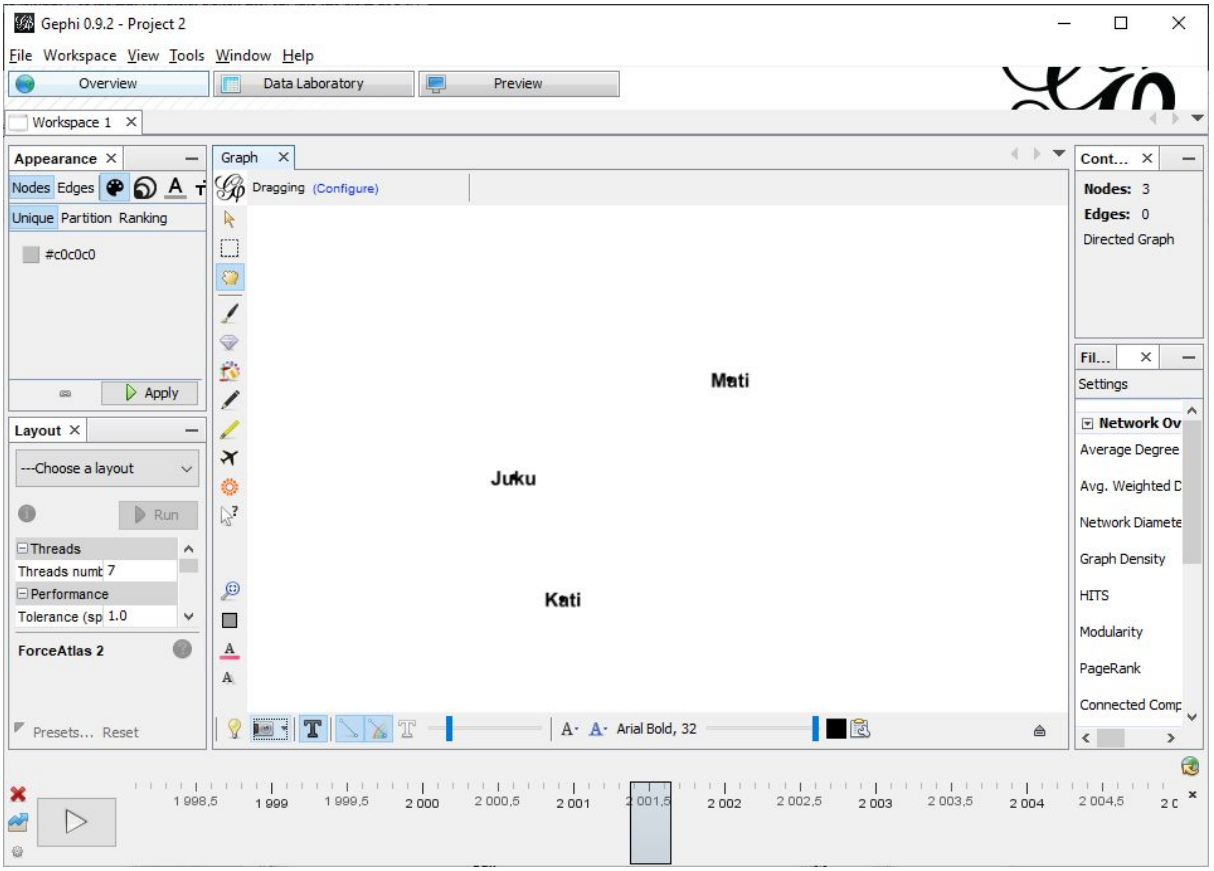
Et Gephi võtaks aastaid ajavahemikuna, tuleb pärast andmete sisse saamist "alates" ja "kuni" tulp ühendada üheks ajaintervalli tulbaks. Selleks tuleb valida `Merge columns` ning `Join strategy` alt `time interval.`



Selle peale tekib lehe alla rohelise nupuga märk `Enable Timeline`. Sellele vajutades ning Overview peale minnes näeb all ajatelge, kus on võimalik omale sobiv lõik valida. Paistab, et aasta 2000 keskel tegutsesid koos Juku ja Kati.



Uuel aastal lisandus ka Mati



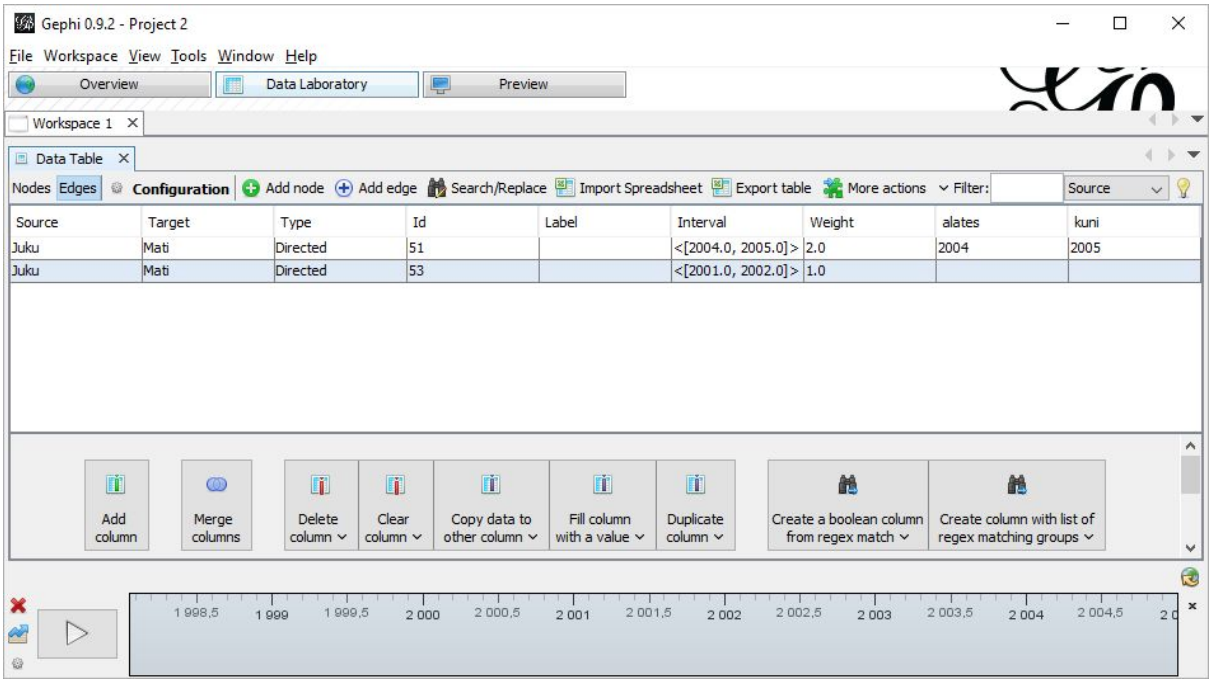
Seoste ajatelg

Graafis on lisaks sõlmedele tähtsal kohal seosed. Ka sinna saab panna ajavahemiku.

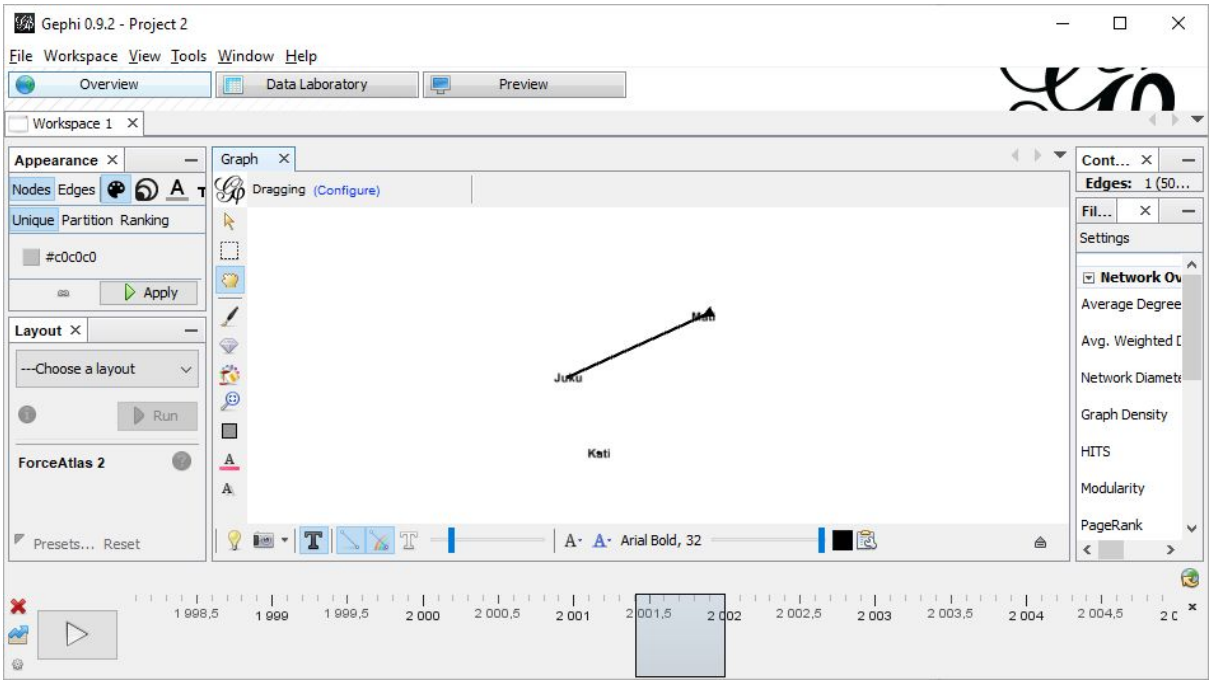
seosed.csv

```
Source,Target,alates,kuni
Juku,Mati,2001,2002
Juku,Mati,2004,2005
```

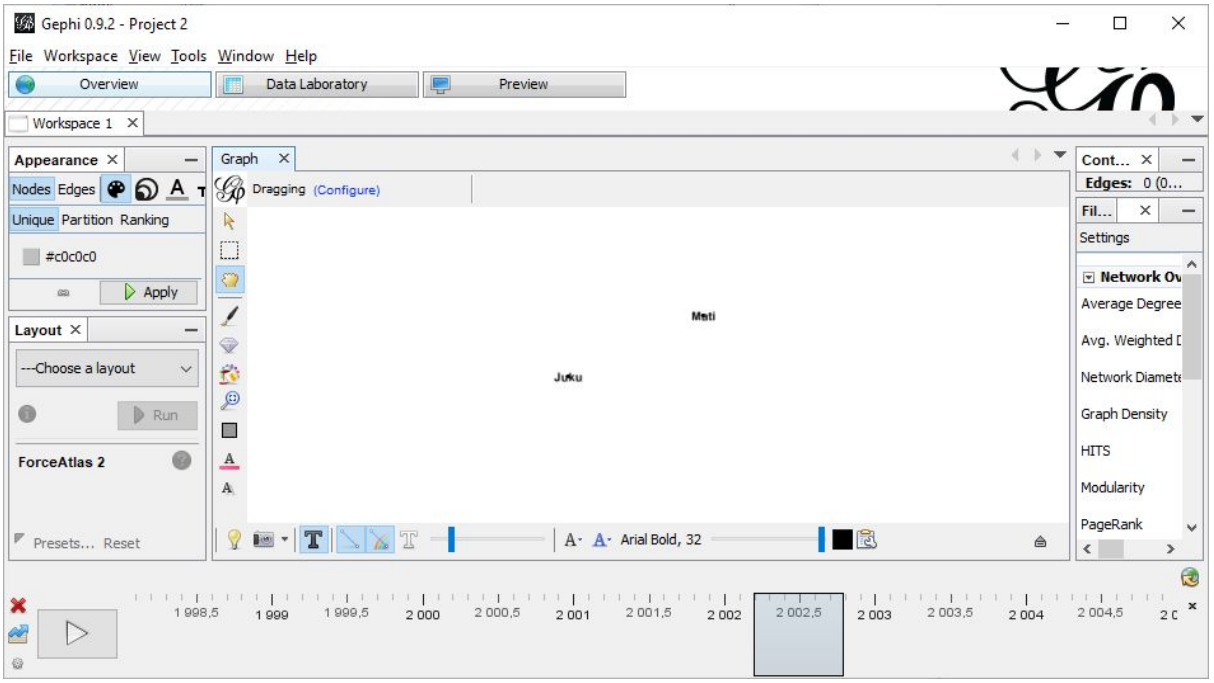
Nagu näha, siis Juku ja Mati tegutsesid koos kahes eraldi lõigus. Samamoodi tulpade ühendamisega saab need ajaintervalliks määrata.



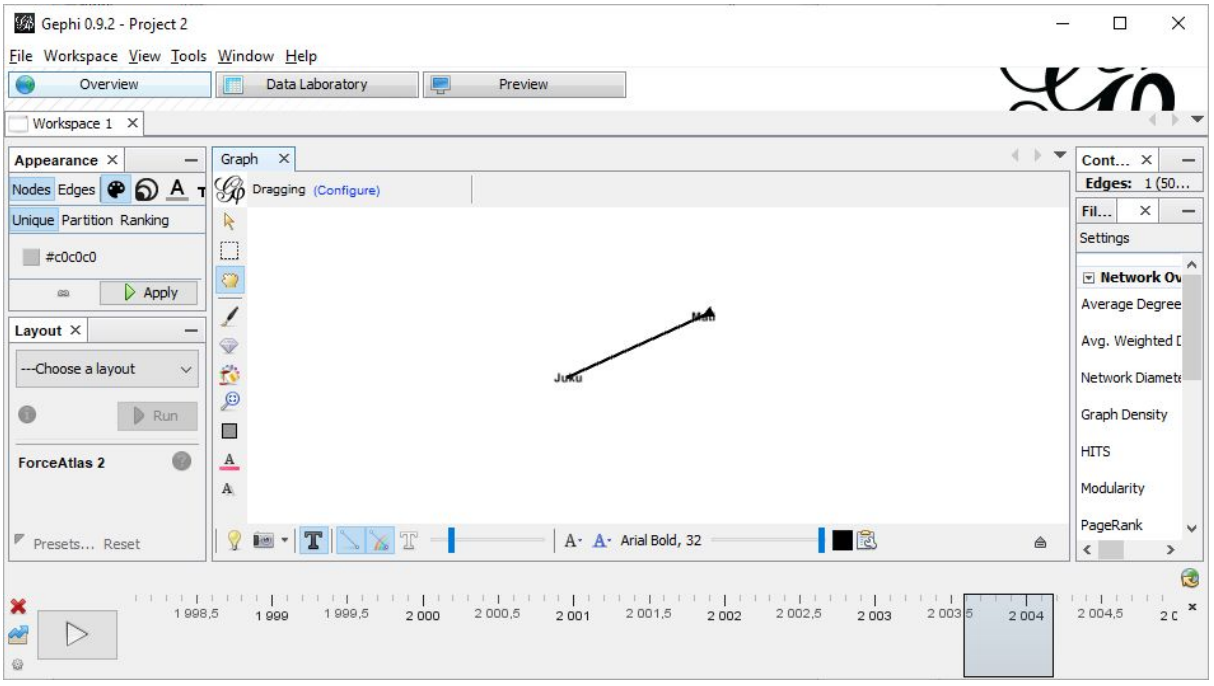
Vastavalt ajaakna paigutusele näeb siis, et kas Juku ja Mati tegutsevad parajasti koos või mitte. 2001. aastal jah



2002. aastal mitte



ning 2004. aastal taas.



Ajavahemikud

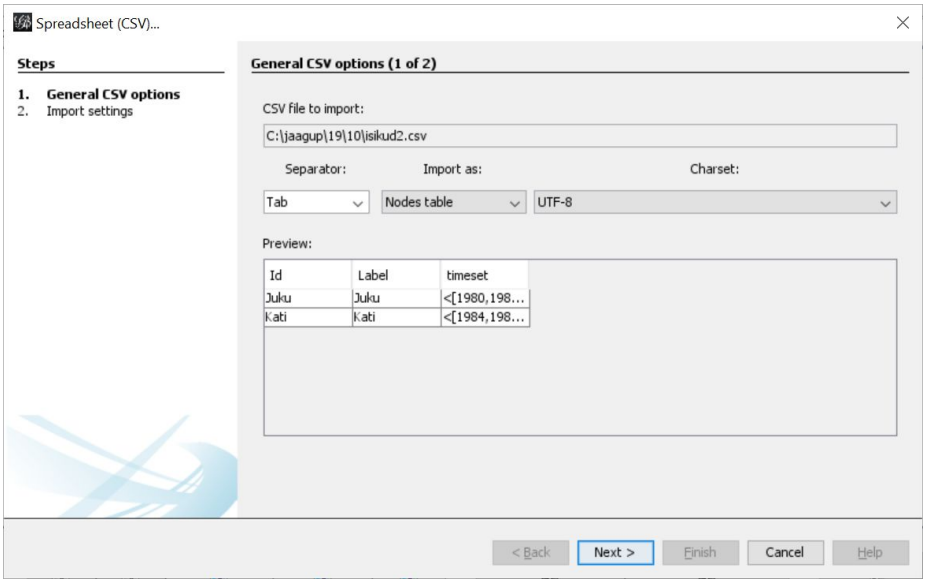
Alguse ja otsa tulbaga saab igale sõlmele määrata vaid ühe ajavahemiku. Sama tegelane võib aga areenil olla korduvalt. Vahepeal ära kaduda ning siis jälle tagasi tulla. Gephis on sarnaste kirjapanekute jaoks tulp nimega timeset (väikese tähega). Noolsulgude <> ning

seal sees omakorda veel ükshaaval semikoolonitega eraldatult kandiliste sulgude vahele pandud aegadega saab määrata, millistel vahemikel on märgitud sõna näha.

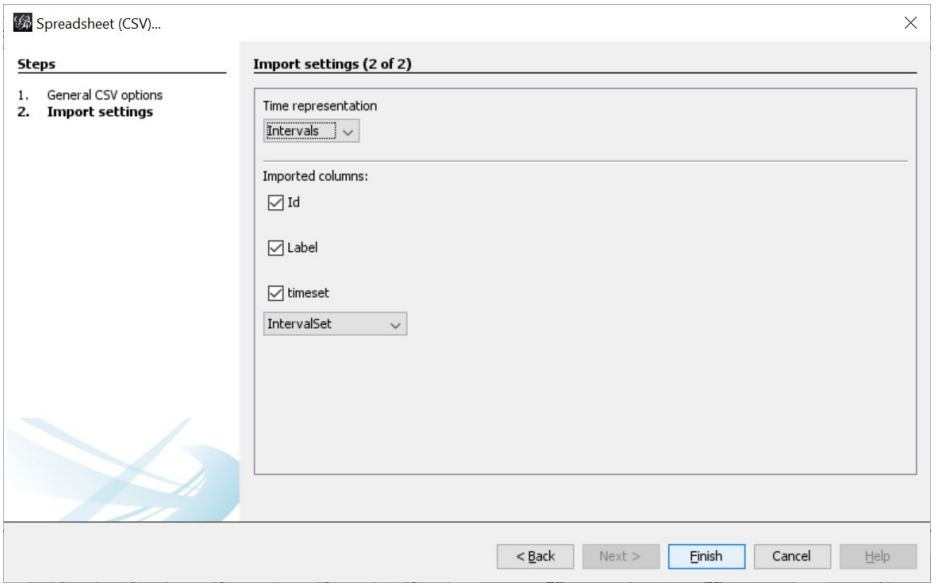
CSV-faili näide:

```
Id      Label   timeset
Juku    Juku    <[1980,1980];[1990,1993]>
Kati    Kati    <[1984,1986];[1992,1992]>
```

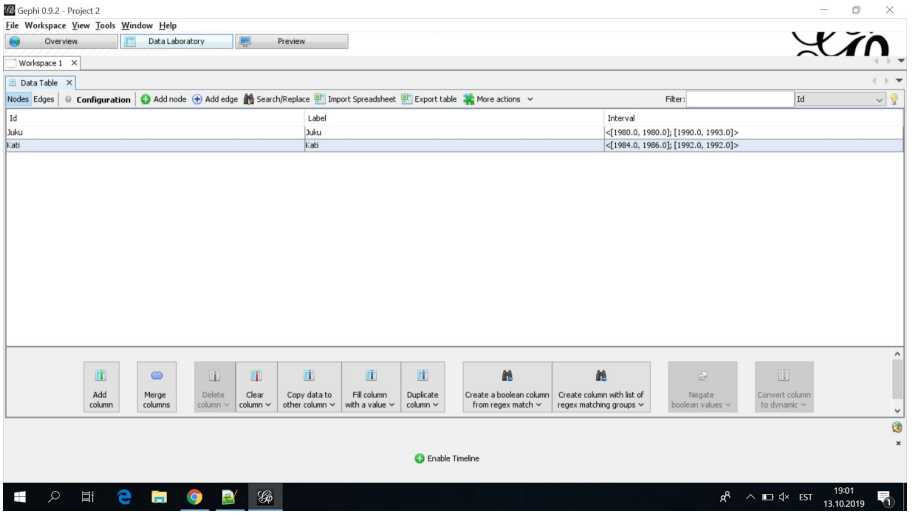
Andmestiku saab lehele sisse lugeda



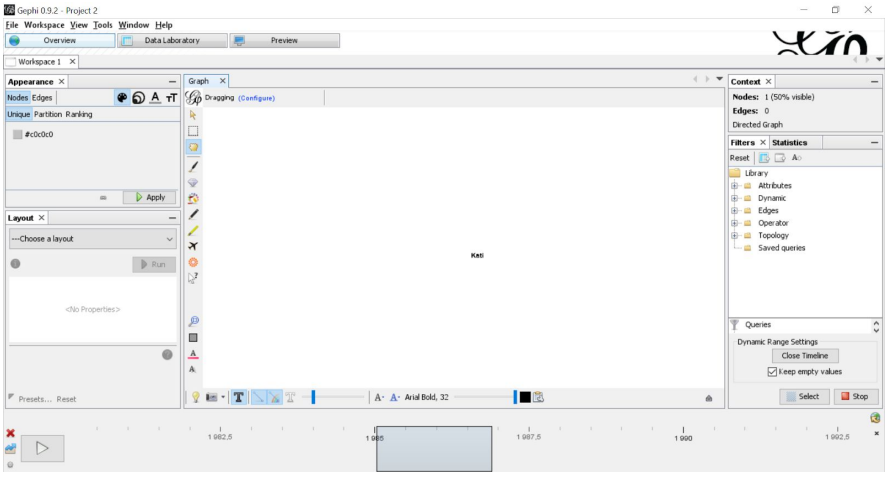
ning siis kinnitada, et just vastavate tulpadega tahetakse tegelda



Sisse loeteluna saab Gephi aru, et tegemist intervalliga.



Edasi on nimi näha vastavalt ajajoonel tehtud valikule.



Harjutus

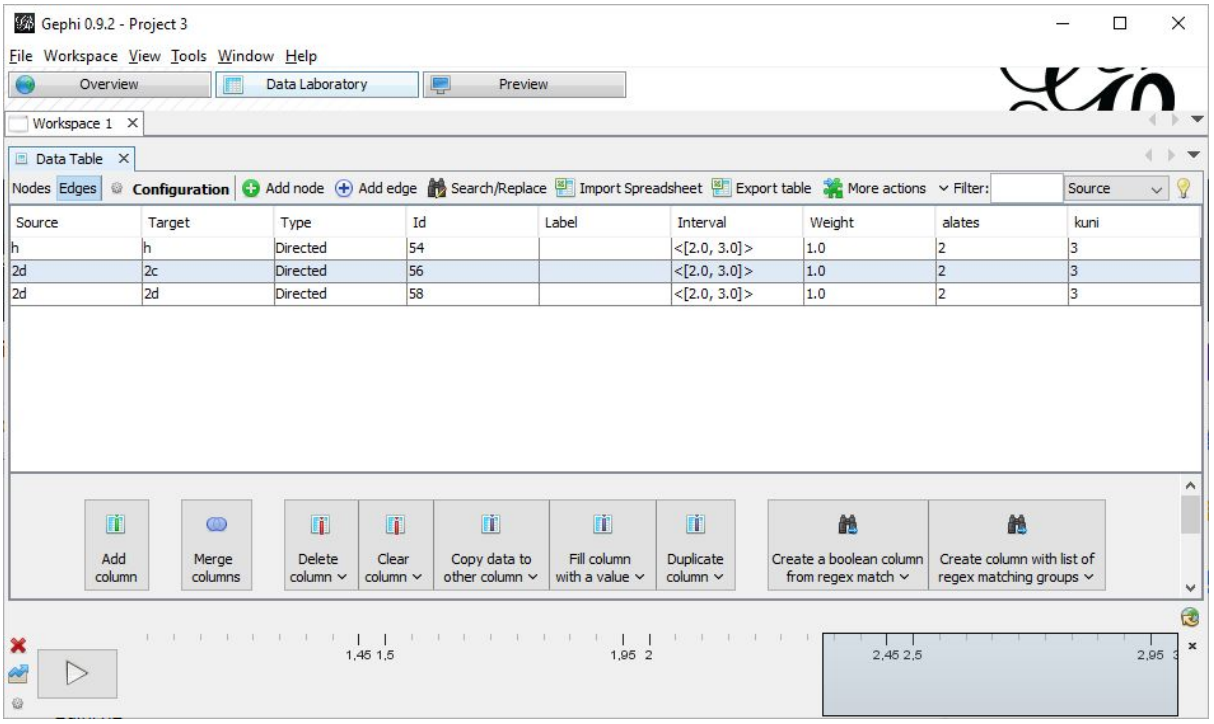
- Tee näide läbi
- Lisa mõned isikud ja ajavahemikud, veendu toimimises

Seosed vastavalt löögile

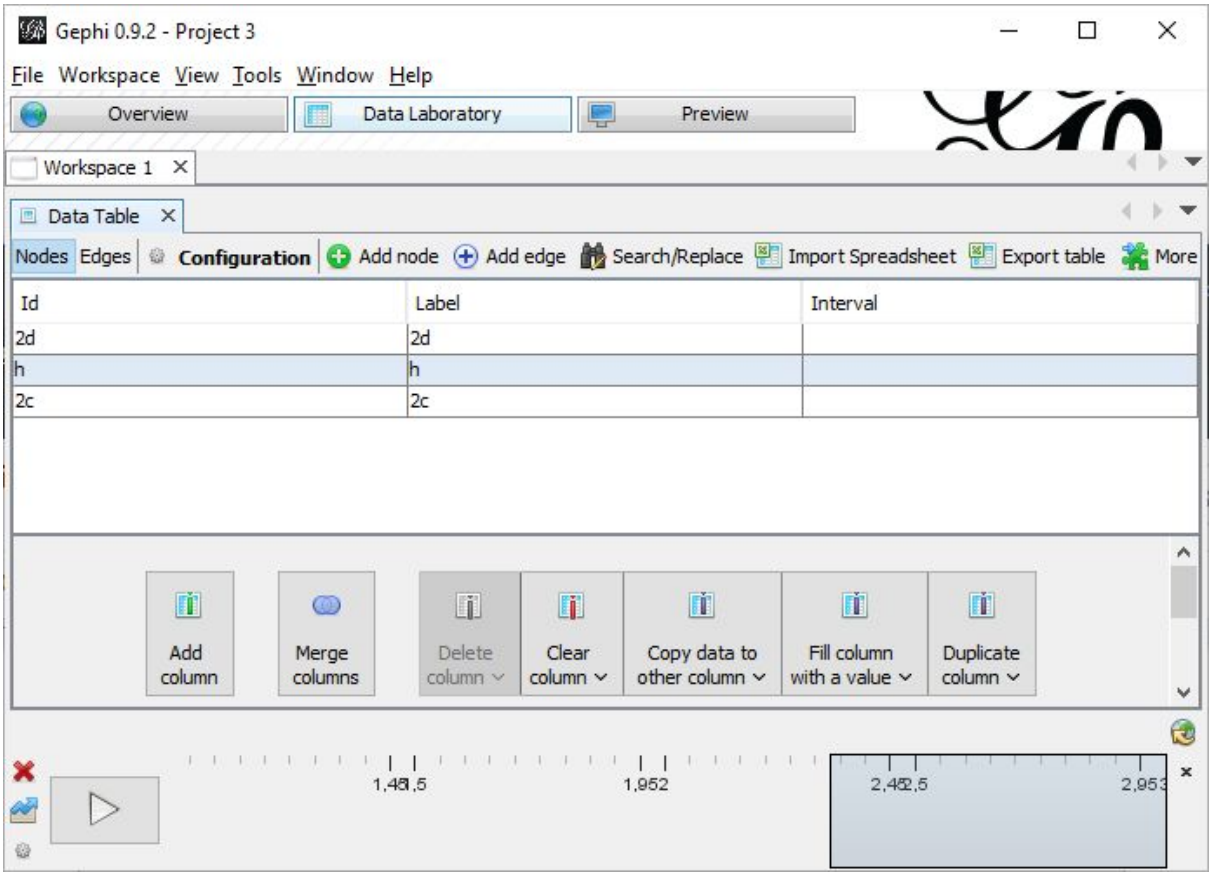
Seosed nootide vahel võivad erineda vastavalt sellele, millise viisi osaga on tegemist. Kuvamiseks koostame kõigepealt noodipaaride tabeli, kus iga paari puhul näha, milliste järjestikuste löökide vahel see moodustatud.

```
Source,Target,alates,kuni
2d,h,1,2
h,h,2,3
2d,2d,1,2
2d,2c,2,3
2d,2d,1,2
2d,2d,2,3
```

Vastavalt ajaaknale on näha aktiivsed seosed



Loetelu sõlmedest ehk nootidest



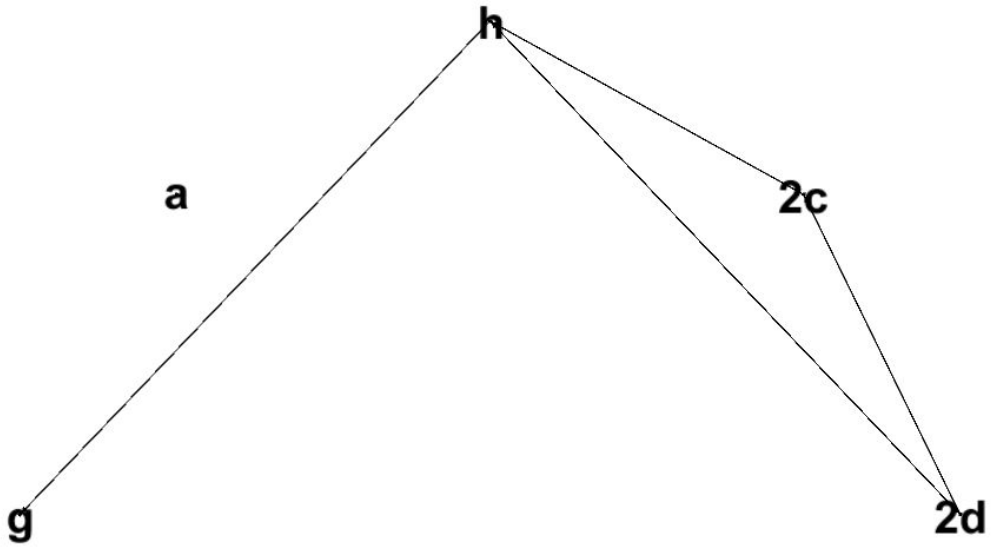
Sarnase tulemuse koostamiseks programmilõik. Üks fail tehakse noodiseostega ning teine nootide koordinaatidega.

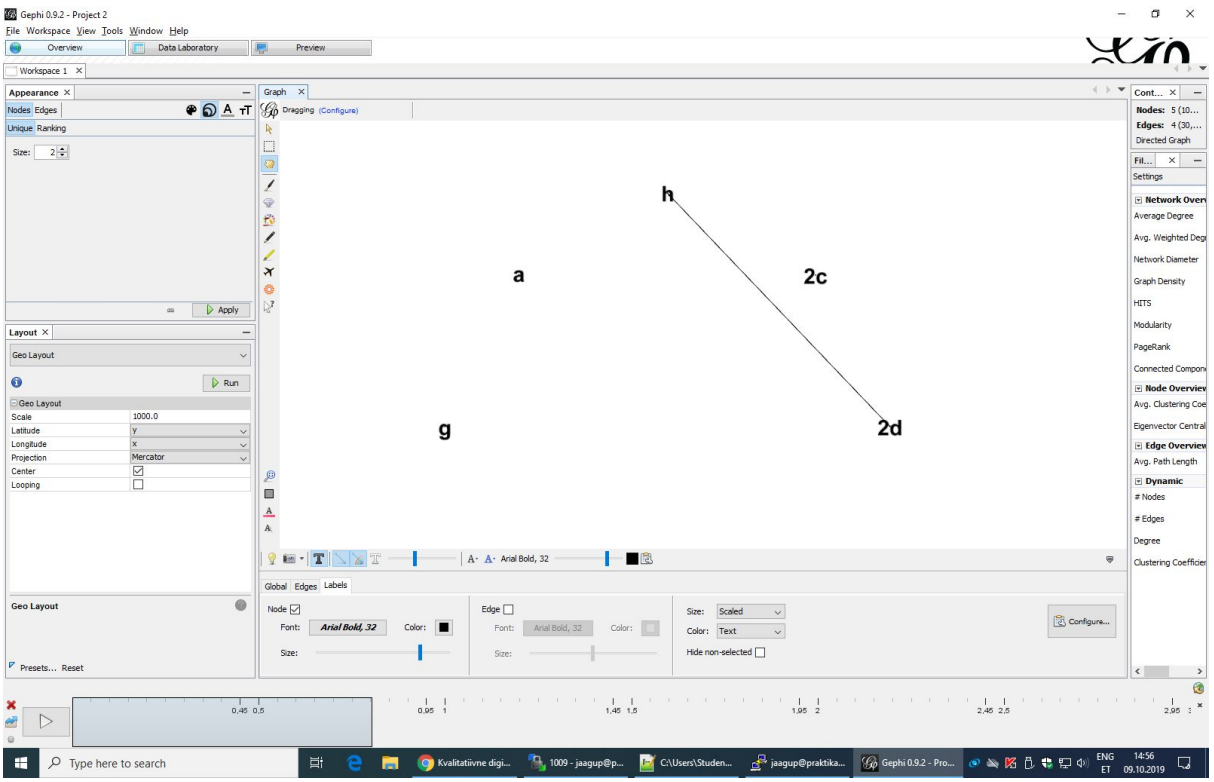
```
import urllib.request
viisid=urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt"
  ).read().decode('utf8').split("\n")[1:]
alles={"g":[0, 0], "a":[10, 20], "h":[30, 30], "2c":[50, 20], "2d":[60, 0]}
f=open("noodiseosed1.csv", "w")
print("Source,Target,alates,kuni", file=f)
nihe=10
pikkus=4
for viis in viisid[:5]:
  m=viis.split(",")
  if m[6]=="g":
    for algus in range(nihe, nihe+pikkus-1):
       if m[algus] in alles and m[algus+1] in alles:
         print(m[algus]+","+m[algus+1]+","+str(algus-nihe)+","+str(algus-nihe+1), file=f)
f.close()


f2=open("noodid1.csv", "w")
print("Id,Label,x,y", file=f2)
for noot in alles:
  print(noot+","+noot+","+str(alles[noot][0])+","+str(alles[noot][1]), file=f2)
f2.close()
```

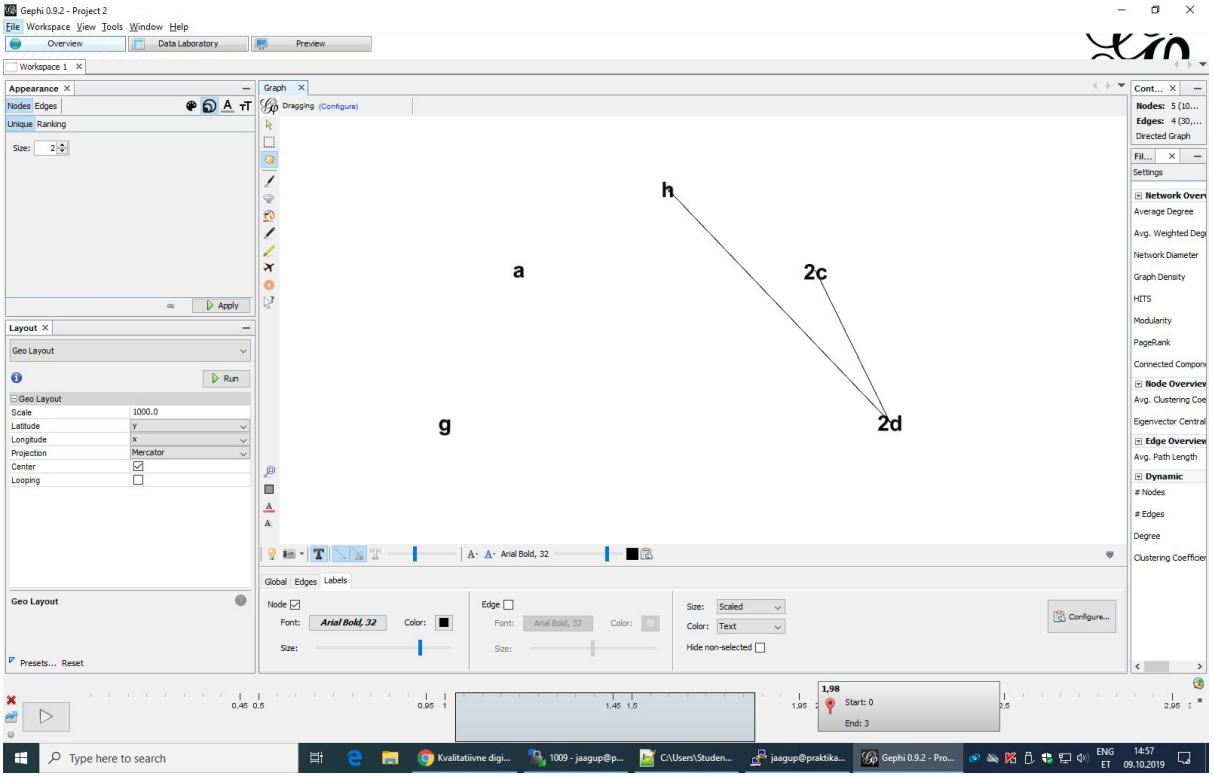
Valminud failid:

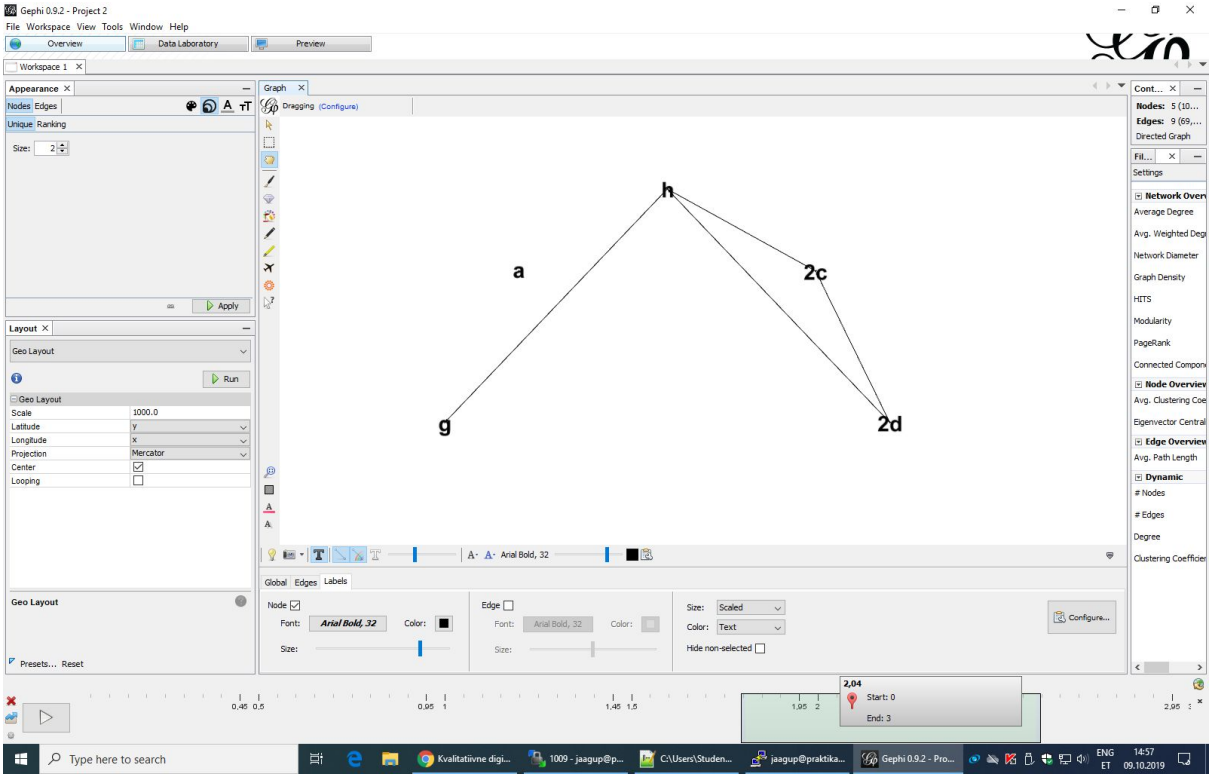
```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1009 $ more
noodiseosed1.csv
Source,Target,alates,kuni
2d,h,0,1
h,h,1,2
h,g,2,3
2d,2d,0,1
2d,2c,1,2
2c,h,2,3
2d,2d,0,1
2d,2d,1,2
2d,2c,2,3
2c,2c,2,3
h,2d,0,1
2d,h,1,2
h,2c,2,3

jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1009 $ more noodid1.csv
Id,Label,x,y
2c,2c,50,20
2d,2d,60,0
a,a,10,20
h,h,30,30
g,g,0,0
```

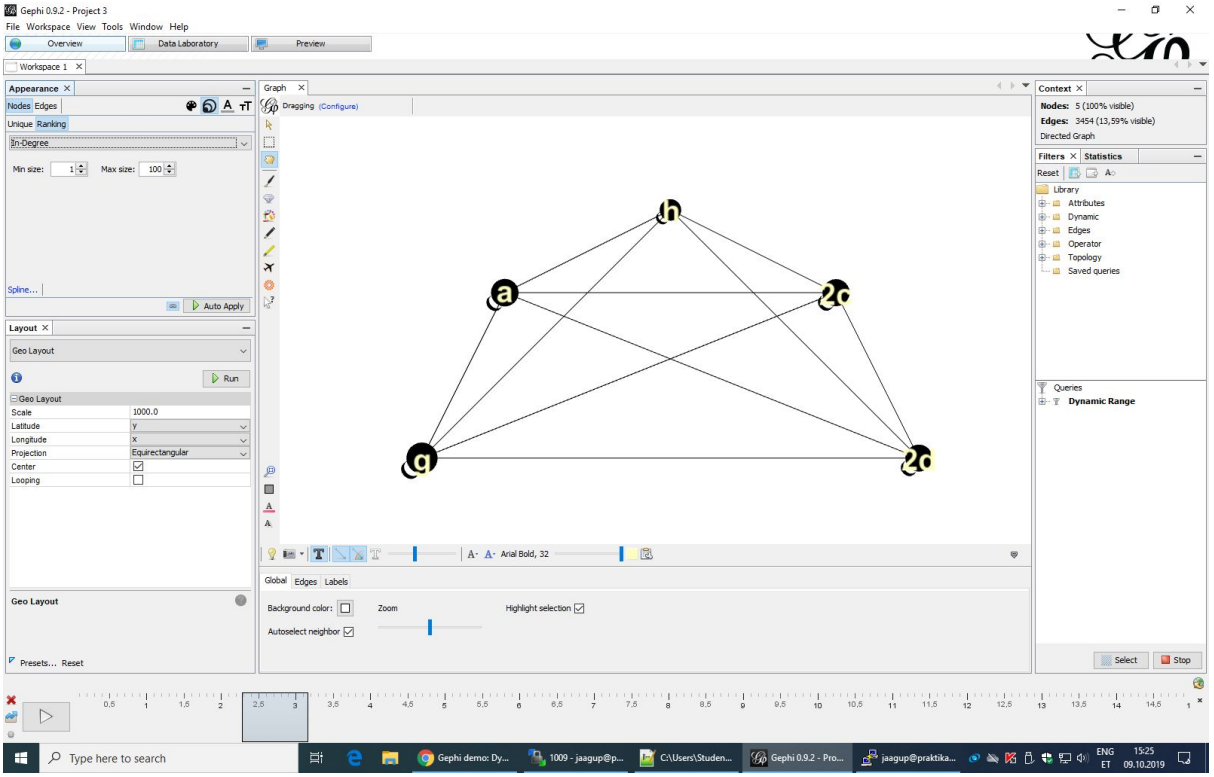
Joonis vastavalt taktiaknale

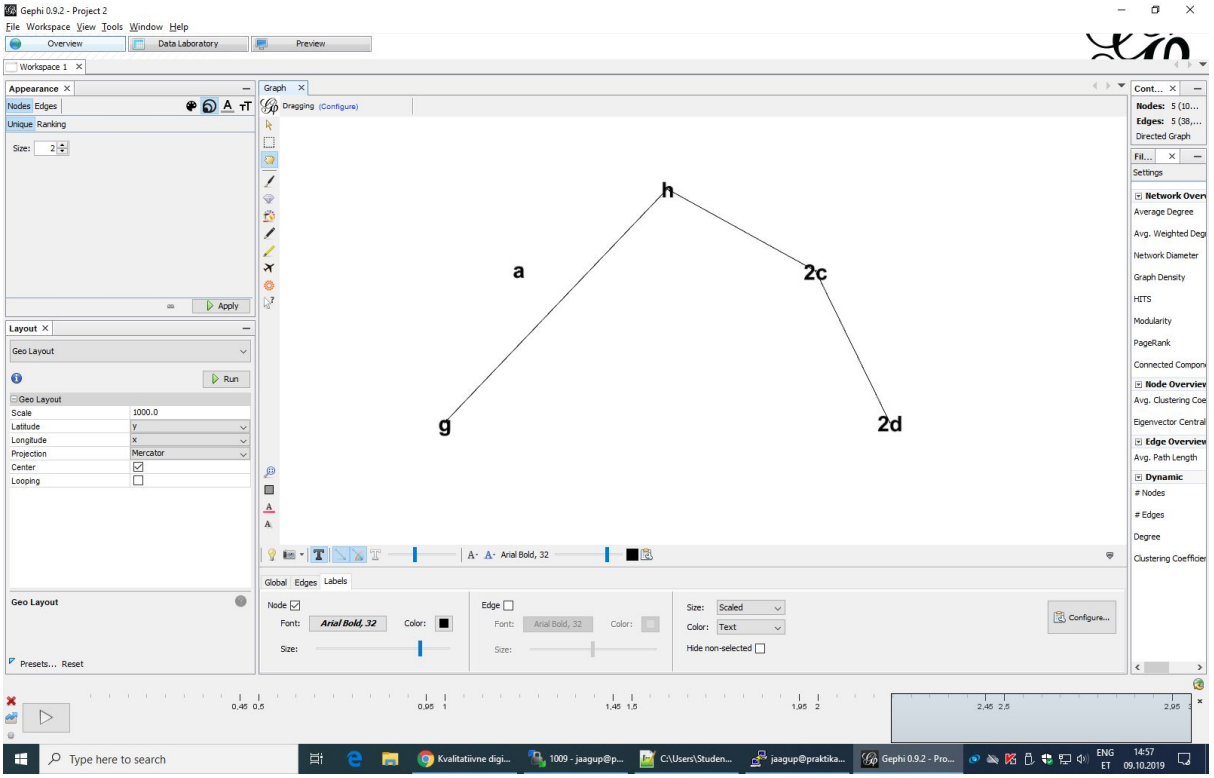

h
a
2c
g
2d


Gephi 0.9.2 - Project 2
File Workspace View Tools Window Help
Overview
Data Laboratory
Preview
Workspace 1
Appearance
Nodes Edges
Unique Ranking
Size: 2
Apply
Layout
Geo Layout
Run
Geo Layout
Scale 1000.0
Latitude
Longitude
Projection Mercator
Center
Looping
Presets... Reset
Graph
Dragging (Configure)
h
a
2c
g
2d
Arial Bold, 32
Global Edges Labels
Node
Font: Arial Bold, 32
Color:
Size:
Edge
Font: Arial Bold, 32
Color:
Size:
Size: Scaled
Color: Text
Hide non-selected
Configure...
Cont...
Nodes: 5 (10...
Edges: 4 (30,...
Directed Graph
Fil...
Settings
Network Overv
Average Degree
Avg. Weighted Degr
Network Diameter
Graph Density
HITS
Modularity
PageRank
Connected Compon
Node Overview
Avg. Clustering Coe
Eigenvector Central
Edge Overview
Avg. Path Length
Dynamic
Nodes
Edges
Degree
Clustering Coefficie
Type here to search
Kvalitatiivne digi...
1009 - jaagup@p...
C:\Users\Studen...
jaagup@praktika...
Gephi 0.9.2 - Pro...
ENG
ET
14:56
09.10.2019


Gephi 0.9.2 - Project 2
File Workspace View Tools Window Help
Overview
Data Laboratory
Preview
Workspace 1
Appearance
Nodes Edges
Unique Ranking
Size: 2
Apply
Layout
Geo Layout
Run
Geo Layout
Scale 1000.0
Latitude y
Longitude x
Projection Mercator
Center
Looping
Geo Layout
Presets... Reset
Graph
Dragging (Configure)
h
a
2c
g
2d
Arial Bold, 32
Global Edges Labels
Node
Font: Arial Bold, 32
Color:
Size:
Edge
Font: Arial Bold, 32
Color:
Size:
Size: Scaled
Color: Text
Hide non-selected
Configure...
Cont...
Nodes: 5 (10...
Edges: 4 (30,...
Directed Graph
Fil...
Settings
Network Overv
Average Degree
Avg. Weighted Degr
Network Diameter
Graph Density
HITS
Modularity
PageRank
Connected Compon
Node Overview
Avg. Clustering Coe
Eigenvector Central
Edge Overview
Avg. Path Length
Dynamic
Nodes
Edges
Degree
Clustering Coefficie
1,98
Start: 0
End: 3
Type here to search
ENG
ET
14:57
09.10.2019


Gephi 0.9.2 - Project 2
File Workspace View Tools Window Help
Overview
Data Laboratory
Preview
Workspace 1
Appearance
Nodes Edges
Unique Ranking
Size: 2
Apply
Layout
Geo Layout
Run
Geo Layout
Scale 1000.0
Latitude y
Longitude x
Projection Mercator
Center
Looping
Geo Layout
Presets... Reset
Graph
Dragging (Configure)
h
a
2c
g
2d
Arial Bold, 32
Global Edges Labels
Node
Font: Arial Bold, 32
Color:
Size:
Edge
Font: Arial Bold, 32
Color:
Size:
Size: Scaled
Color: Text
Hide non-selected
Configure...
Cont...
Nodes: 5 (10...
Edges: 9 (69,...
Directed Graph
Fil...
Settings
Network Overv
Average Degree
Avg. Weighted Degr
Network Diameter
Graph Density
HITS
Modularity
PageRank
Connected Compon
Node Overview
Avg. Clustering Coe
Eigenvector Central
Edge Overview
Avg. Path Length
Dynamic
Nodes
Edges
Degree
Clustering Coefficie
2,04
Start: 0
End: 3
Type here to search
ENG
ET
14:57
09.10.2019


Gephi 0.9.2 - Project 3
h
a
2c
g
2d


Gephi 0.9.2 - Project 2
h
a
2c
g
2d

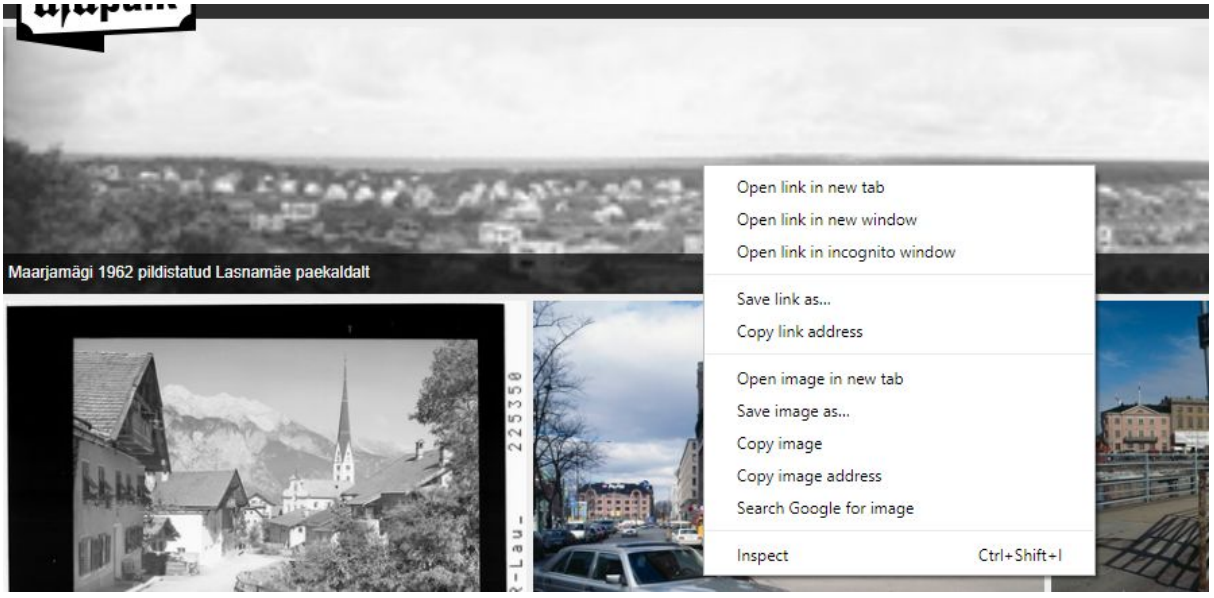
Harjutus

- Pane näited tööle
- Koosta sarnane muudetav joonis a-ga algavate neljatäheliste sõnade tähtedevaheliste seoste kohta

Veebiteenused

Rakendusi soovitakse mõnikord omavahel ühendada - kõike ei saa ja pea alati ise tegema ning mõnigikord on vaja välisest allikast iga mõne aja tagant uusi ja värskeid andmeid, et neid näiteks enese omadega võrdlusena kuvada. Sageli on võõras rakendus kirjutatud teistsuguses keeles, asub turvapiirangute taga või muul puhul ei pääse sellele kuigi lihtsalt ligi. Tänapäeval toimib märgatav osa lahendusi veebis ning veebiteenused on üks mugav moodus võõrsilt omale killukeste juurde haakimiseks ning vajalike arvutuste ja lisanduste teha laskmiseks. Milline on lahenduse oma poole ning võõrsilt juurde hangitud võimaluste vahekord, see sõltub juba loodavast lahendusest.

Oma lehele väljapoolt pildi lisamiseks tuleb kõigepealt kindlaks teha lisatava pildi aadress. Veebiseiluri parem hiireklõps aitab sellele vahel kaasa.



Edasi võimalik see aadress oma html-faili lisada

```
<!doctype html>
<html>
   <head>
      <title>Pildid</title>
         <style>
```

```
            body{background-color: #cfffcc}
         </style>
   </head>
   <body>
      <h1>Piltide leht</h1>
         <img src="https://ajapaik.ee/photo-thumb/187788/400/" />
   </body>
</html>
```

ja lehte vaadata

Leht lehes

Teise lehe lisamiseks oma lehe sisse sobib `iframe` käsklus

```
<!doctype html>
<html>
   <head>
      <title>Pildid</title>
         <style>
           body{background-color: #cfffcc}
         </style>
         <meta charset="utf-8" />
   </head>
   <body>
      <h1>Piltide leht</h1>
         <img src="https://ajapaik.ee/photo-thumb/187788/400/" /> <br />

         Pilt võetud lehestikust <a href="http://ajapaik.ee/">ajapaik.ee</a> <br />
         Leht koostatud kursusel: <br />
         <iframe
src="http://minitorn.tlu.ee/~jaagup/kool/java/kursused/19/kvalitatiivne_digihumanitaaria/juht.html"></iframe>
   </body>
</html>
```

Tulemusena ongi üks leht teise lehe sees. Nõnda võib lisada mitmesuguseid valmislahendusi - graafikuid, tabeleid, mänge ja muid rakendusi.

Harjutus

- Tee näide läbi
- Koosta ise väike veebileht, kus sees tekst, pilt ning teine veebilehekülg

JSON

Mitmekülgsete andmete üheks sobivaks ülekandevormiks on JSON (JavaScript Object Notation). Seal on iga väärtuse juurde omadusena kirjutatud, et millega tegemist. Üks JSONi näide ajapaik.ee lehelt pildi andmete küsimise kohta. Aadress:

```
https://opendata.ajapaik.ee/photos/148348/geotags/?format=json
```

Ja tulemus


```
[{"id":215856,"lat":59.4362959945468,"lon":24.7426373215637,"geography":"SRID=4326;POINT
(24.7426373215637
59.4362959945468)","azimuth":298.996843617767,"azimuth_line_end_lat":59.4364662079086,"azim
uth_line_end_lon":24.7423302086315,"zoom_level":18,"origin":0,"type":0,"map_type":2,"hint_u
sed":false,"photo_flipped":false,"is_correct":true,"azimuth_correct":true,"score":391,"azim
uth_score":100,"trustworthiness":0.973063973063973,"created":"2018-09-23T06:56:15.996981Z",
"modified":"2018-09-23T06:56:17.216411Z","user":1150476,"photo":148348}]
```

Välimised kantsulud näitavad, et tuleb andmete loetelu (mis praegu koosneb vaid ühest kirjest). Edasi looksulgude vahel kirje omadused, kätte püüame saada laius- ja pikkuskraadi. Käsklus `json_decode` suudab PHP puhul JSON-failist sobiva objekti tekitada, mille seest siis edasi andmeid küsida.

```
<?php

$obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/148348/geotags/?format=json"));
 echo $obj[0]->lat."<br />".$obj[0]->lon;
```

Tulemus:

```
59.436295994547
24.742637321564
```

Foto üldisemad andmed saab ilma parameetrit `geotags` lisamata.

```
https://opendata.ajapaik.ee/photos/148348/?format=json
```

Tulemuse näide:


```
{"id":148348,"rephotos":[],"similar_photos":[],"geotags":"https://opendata.ajapaik.ee/photos/148348/geotags/","image":"https://opendata.ajapaik.ee/media/uploads/muis_sLh7NnQ.jpg","image_unscaled":null,"image_no_watermark":null,"height":558,"width":800,"aspect_ratio":null,"flip":false,"invert":false,"stereo":false,"rotated":null,"date":null,"date_text":null,"title":null,"title_et":null,"title_en":null,"title_ru":null,"title_fi":null,"title_sv":null,"title_nl":null,"title_de":null,"title_no":null,"description":"Tema Pühadus paavst Johannes Paulus II oma legendaarsel kuulikindlast klaasist pealisehitisega Mercedesel läbi linna sõitmas.","description_et":"Tema Pühadus paavst Johannes Paulus II oma legendaarsel kuulikindlast klaasist pealisehitisega Mercedesel läbi linna sõitmas.",  …
```


Andmete kuvamine lehel:

```
<?php

$obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/148348/?format=json"));
 echo $obj->description;
```

Tulemus:

Tema Pühadus paavst Johannes Paulus II oma legendaarsel kuulikindlast klaasist pealisehitisega Mercedesel läbi linna sõitmas.

Harjutus

- Pane näide PHP-võimelises kohas tööle

- Küsi andmed välja fotolt aadressiga `https://opendata.ajapaik.ee/photos/198312/?format=json`

Leht vastavalt parameetrile

Oma lehele saab samuti sobivaid andmeid ette anda - praegusel juhul kuvatava foto numbri. See omakorda pannakse veebiaadressile, kust pilti vaadatakse.

```
<?php
 $nr=148348;
 if(isset($_REQUEST["nr"])){$nr=intval($_REQUEST["nr"]);}

$obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/$nr/?format=json"));
 echo $obj->description;
```

```
http://praktika1.cs.tlu.ee/~jaagup/2019/kvalitatiivne_digihumanitaaria/1016/json3.php?nr=100000
```

Väljund:

Linnavaade. Marja tn 30 Tartus 1992a.

Harjutus

- Koostage veebileht, kus aadressireale pildinumbri sisestamisega kuvatakse ajapaik.ee serverist vastava koodiga pildi kirjeldus ja koordinaadid
- Lisa võimalusel ka pilt ise leheküljele

```
<?php
 $nr=148348;
 if(isset($_REQUEST["nr"])){$nr=intval($_REQUEST["nr"]);}
 $obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/$nr/?format=json"));
 echo $obj->description."<br />";


 $obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/$nr/geotags/?format=json"));
 echo $obj[0]->lat."<br />".$obj[0]->lon;
```

Avamine:

```
http://praktika1.cs.tlu.ee/~jaagup/2019/kvalitatiivne_digihumanitaaria/1016/json4.php?nr=100000
```

Tulemus:

Linnavaade. Marja tn 30 Tartus 1992a.
58.387397972468
26.713786780595

Andmete korjamisel muutujatesse saab neid edasi kergemini lehe sees välja näidata.

```
<?php
 $nr=148348;
 if(isset($_REQUEST["nr"])){$nr=intval($_REQUEST["nr"]);}
 $obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/$nr/?format=json"));
 $kirjeldus=$obj->description;
 $pildiaadress=$obj->image;

 $obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/$nr/geotags/?format=json"));
 $asukoht=$obj[0]->lat."<br />".$obj[0]->lon;
?>
<!doctype html>
<html>
 <head>
   <title>Pildi <?php echo $nr; ?> leht</title>
 </head>
 <body>
    Kirjeldus: <?php echo $kirjeldus; ?><br />
         Asukoht: <br /><?php echo $asukoht; ?><br />
         <img src="<?php echo $pildiaadress; ?>">
 </body>
</html>
```

Vaatamine:

`http://praktika1.cs.tlu.ee/~jaagup/2019/kvalitatiivne_digihumanitaaria/1016/json5.php?nr=100000`

Kirjeldus: Linnavaade. Marja tn 30 Tartus 1992a.
Asukoht:
58.387397972468
26.713786780595

Failid võivad ka edasi näidata. Siin saadakse kõigepealt opendata.ajapaik.ee avalehelt piltide loetelu viidete ja pealkirjadega ning edasi pildile vajutades kuvatakse eraldi lehel vastav pilt koos andmetega.

```
<?php
  $obj=json_decode(file_get_contents("https://opendata.ajapaik.ee/photos/?format=json"));
```

```
?>
<!doctype html>
<html>
  <head>
    <title>Piltide andmete loetelu</title>
  </head>
  <body>
    <ul>
         <?php foreach($obj->results as $pilt){
              echo "<li><a href='json5.php?nr=$pilt->id'>$pilt->description</a></li>";
         }
         ?>
       </ul>
  </body>
</html>
```

Tulemus:

- Maarjamägi 1962 pildistatud Lasnamäe paekaldalt
- Axams (878m) gegen Nordkette (Tirol)
- Mikonkatu 9,11,13,15,17. Etualalla ravintola Planet Hollywood.
- Kauppatori, kävelysilta Keisarinluodonlaiturille. Taustalla Pohjoisesplanadi 1,3,5,7,9.
- Pitkänsillanranta ja Pitkäsilta. Uusi kivisilta valmistui 1912 puisen tilalle.
- Kaivokatu 9, 7 (= Asema-aukio). Hillerin korttelin taloja, jotka purettiin uuden rautatieaseman tieltä.
- Kaivokatu 12,10. Etualalla Helsingin Kaupungin Liikennelaitoksen infotaulu.
- Rautatientori, oikealla Mikonkatu 15,17,19.
- Helsingin rautatieasema Kaivokadun puolelta nähtynä.
- Mannerheimintie 9, Kaivokadun ja Mannerheimintien risteys.

Kirjeldus: Mikonkatu 9,11,13,15,17. Etualalla ravintola Planet Hollywood.
Asukoht:
60.169679274681
24.945321321487

RSSi lugemine

Uudistevorming on aastaid populaarne moodus andmete edastamiseks. Näitena Postimehe RSS
https://www.postimees.ee/rss

```
<?xml version="1.0" encoding="utf-8"?>
 <rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
```

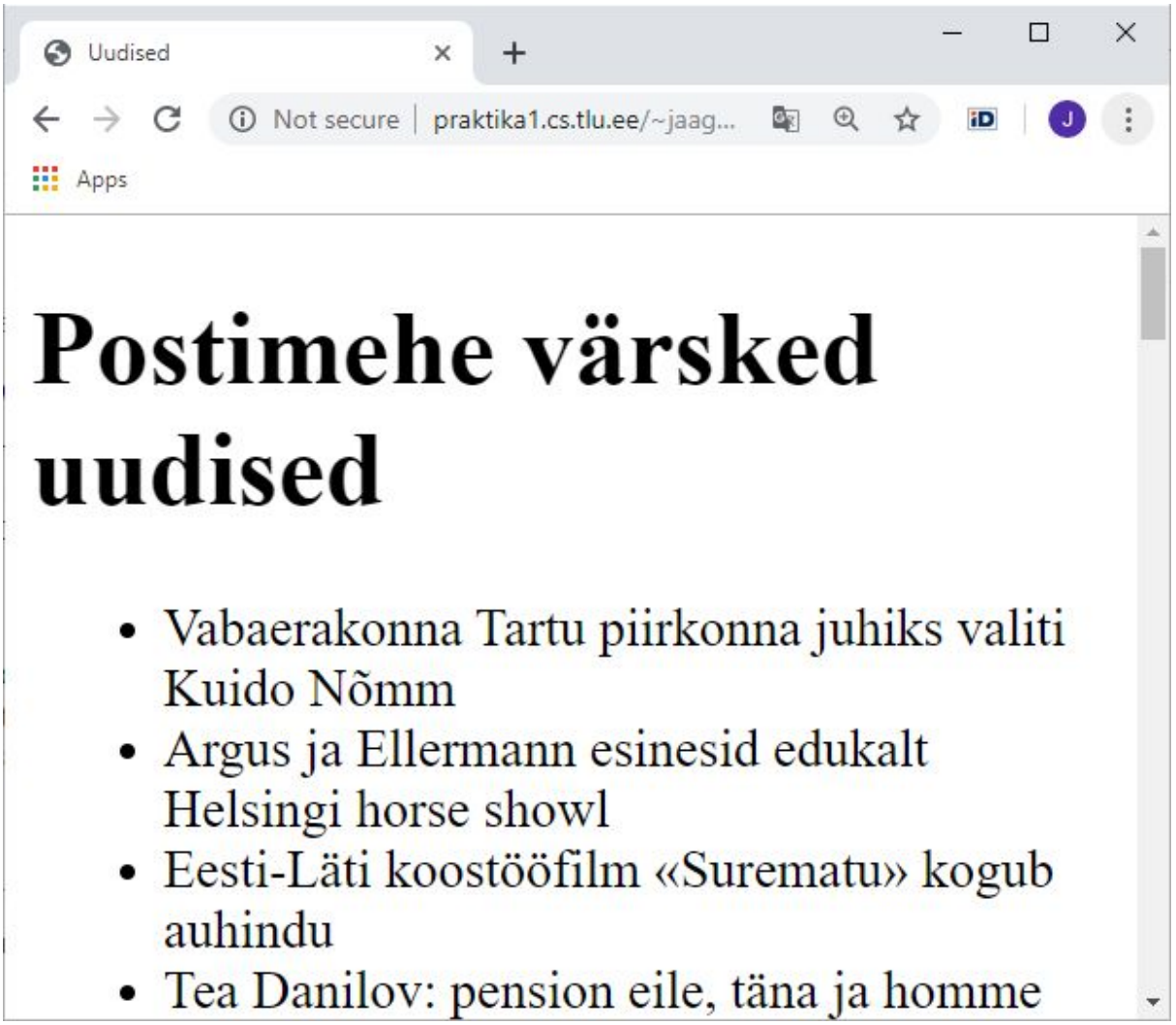
```
  <channel>
  <atom:link href="https://www.postimees.ee/rss" rel="self" type="application/rss+xml"/>
  <generator>RSS Generator 2.1.1</generator>
  <docs>http://www.rssboard.org/rss-specification</docs>
  <title>Postimees</title>
  <language>et</language>
  <pubDate>Mon, 28 Oct 2019 13:12:36 +0000</pubDate>
  <link>https://www.postimees.ee</link>
  <description>Postimees: Värsked uudised Eestist ja välismaalt. Loe lähemalt</description>
  <image>
   <url>https://f.pmo.ee/logos/81/6f1c8d9387ad9a7bfa5c20eb59287380.svg</url>
   <title>Postimees</title>
   <link>https://www.postimees.ee</link>
   <width>230</width>
   <height>45</height>
   <description>Postimees: Värsked uudised Eestist ja välismaalt. Loe lähemalt</description>
  </image>
  <item>
   <title>Argus ja Ellermann esinesid edukalt Helsingi horse showl</title>
   <description>Nädalavahetusel Helsingis toimunud nimekal horse show'l võistlesid maailmatippude kõrval ka
Paul-Richard Argus ja Dina Ellermann, kes naasesid võistluselt mitmete võitude ja auhinnaliste
kohtadega.</description>
   <link>https://pmo.ee/6812233</link>
   <guid isPermaLink="false">pm#6812233</guid>
   <pubDate>Mon, 28 Oct 2019 15:10:58 +0200</pubDate>
   <author>Eesti Ratsaspordi Liit</author>
   <category domain="https://sport.postimees.ee">Sport</category>
   <category domain="https://sport.postimees.ee/section/3127">Ratsutamine</category>
   <category domain="https://sport.postimees.ee/section/3129">Takistussõit</category>
   <enclosure
url="https://f7.pmo.ee/xysA0Yfq2yRLcm8oEuYFQ-XgOJ8=/480x270/smart/nginx/o/2019/03/24/11892960t1hbe27.jpg"
length="" type="image/jpeg"/>
  </item>
  <item>
   <title>Eesti-Läti koostööfilm «Surematu» kogub auhindu</title>
   <description>Juuli alguses Karlovy Vary A-klassi festivalil esilinastunud ja dokumentaalfilmi Grand Prix
võitnud «Surematu» võitis ühe nädala jooksul kaks peaauhinda. Ksenija Ohhapkina Oscarile kandideeriv
dokumentaalfilm tunnistati parimaks Astra rahvusvahelisel festivalil Rumeenias ja nädal hiljem ArtDokfesti
võistlusprogrammis Riia filmifestivalil.</description>
   <link>https://pmo.ee/6812232</link>
   <guid isPermaLink="false">pm#6812232</guid>
   <pubDate>Mon, 28 Oct 2019 15:07:41 +0200</pubDate>
   <author>Kultuuritoimetus</author>
   <category domain="https://kultuur.postimees.ee">Kultuur</category>
   <category domain="https://kultuur.postimees.ee/section/2193">Film</category>
   <enclosure
url="https://f11.pmo.ee/kip1J4H1NM2FjWP_XxByt0spJxw=/480x270/smart/nginx/o/2019/09/08/12542394t1h76bc.jpg"
length="" type="image/jpeg"/>
  </item>
  <item>
   <title>Tea Danilov: pension eile, täna ja homme</title>
   <description>Riiklikud pensionisüsteemid, nagu me neid Euroopas tunneme, on mitmete arengute tõttu surve
all, nende «parim enne» hakkab mööda saama, kirjutab Arenguseire Keskuse juhataja Tea Danilov.</description>
   <link>https://pmo.ee/6812226</link>
   <guid isPermaLink="false">pm#6812226</guid>
   <pubDate>Mon, 28 Oct 2019 14:59:48 +0200</pubDate>
   <author>Tea Danilov</author>
   <category domain="https://arvamus.postimees.ee">Arvamus</category>
   <enclosure
url="https://f12.pmo.ee/xdrCh9LaVapKJkPUzEdqCRSTPaE=/480x270/smart/nginx/o/2019/09/25/12595467t1h9c65.jpg"
length="" type="image/jpeg"/>
  </item>
 </channel>
</rss>
```

Sealt saab pealkirjad kätte järgneva koodiga

```
<!doctype html>
```

```
<html>
  <head>
     <title>Uudised</title>
  </head>
  <body>
     <h1>Postimehe värsked uudised</h1>
     <ul>
          <?php
             $juur=simplexml_load_file("https://www.postimees.ee/rss");
                foreach($juur->channel->item as $uudis){
                       echo "<li>$uudis->title</li>";
                }
          ?>
        </ul>
  </body>
</html>
```

Tulemus:

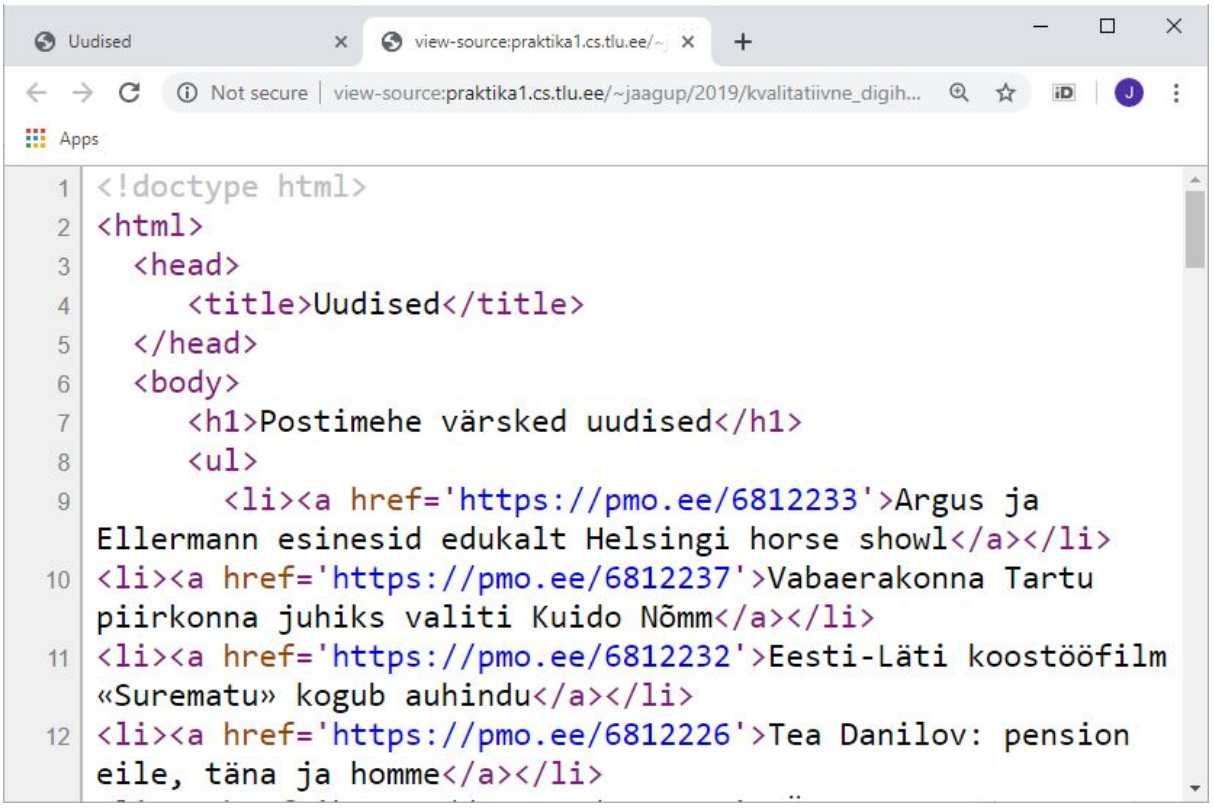


Pealkirjale RSSist vajutatav viide ka juurde

```
<!doctype html>
<html>
```

```
  <head>
     <title>Uudised</title>
  </head>
  <body>
     <h1>Postimehe värsked uudised</h1>
     <ul>
          <?php
             $juur=simplexml_load_file("https://www.postimees.ee/rss");
                foreach($juur->channel->item as $uudis){
                       echo "<li><a href='$uudis->link'>$uudis->title</a></li>\n";
                }
          ?>
        </ul>
  </body>
</html>
```

Trükitavatele ridadele sai juurde ka reavahetus - nii HTMLi lähtekood paremini loetav.



Harjutus

- Uurige, millistelt lehtedelt on võimalik JSON-i, XMLi või muul kujul andmeid kätte saada
- Koostage oma leht, kus kasutatakse mujalt tõmmatud andmeid

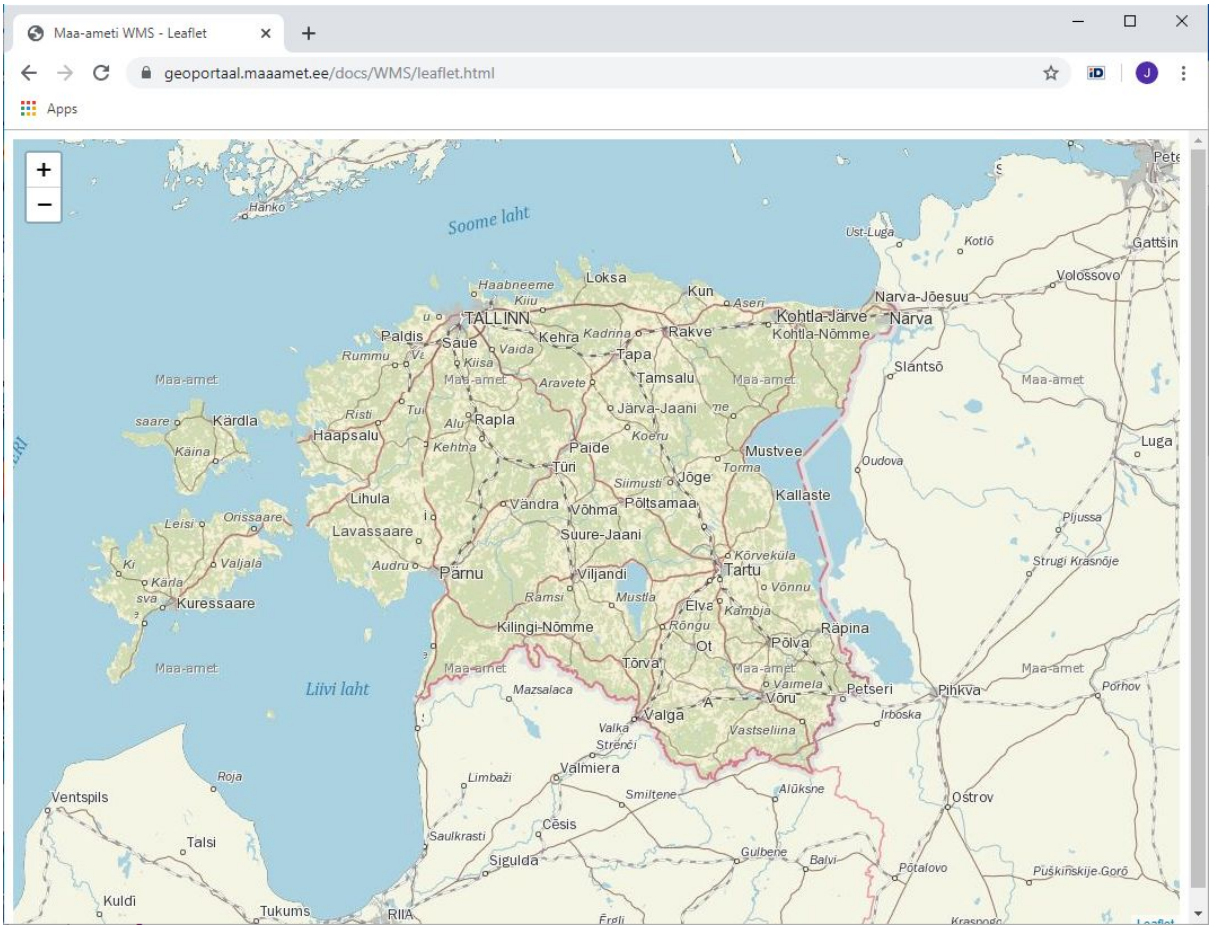
Kaardirakendused

Asukohaandmed toetavad andmestikku mitmesuguste valdkondade juures. Kaardi pealt paistab leidude paiknemine mõnigikord selgemalt välja kui kirjelduste juurest.

Maa-ameti kaardid Leafletiga

Arvutiga hakati üksikuid kaardiandmeid töötlema juba 1960ndatel, vabamalt ja laiemalt ja veebis sai kaardirakendusi kokku panna aga alles pärast sajandivahetust. Maaamet oli selles suhtes usin ning 2005. aasta paiku pääses nende kaudu ligi juba mitmetele kaardikihtidele ning sai kasutada riigipoolseid lahendusi ja luua omi. Maaameti lehe näide:

`https://geoportaal.maaamet.ee/docs/WMS/leaflet.html`



Mugavaks kaardiandmete näitajaks on tõusnud JavaScripti teek Leaflet. Teegil endal kaardiandmeid pole, tuleb näidata eri kohtadest pakutavaid kaarte. Siin näitena maaameti kaardiandmete kuvamine.

Kuvamiseks tuleb luua kaardi objekt

```
kaart = L.map('kaardikiht').setView([59.43, 24.75], 13);
```

Esimese parameetrina näha asukoht (laius- ja pikkuskraad), teisena suurendus. Mida suurem number, seda lähem vaade. Suurenduse 6 juures jääb peale enamvähem terve Euroopa, suurenduse 16 puhul maja ja tänavanurk.

Edasi määrata, milliselt aadressilt ja milliste parameetritega saadakse kätte kaarditükid. Antakse ette suurendus ning asupaiga koordinaadid. Maaameti sooviks on, et parameetril näidataks ära, millise asutuse ja alamosaga on kaartide küsijana tegemist - nii saavad nad hiljem omapoolseid kokkuvõtteid teha.

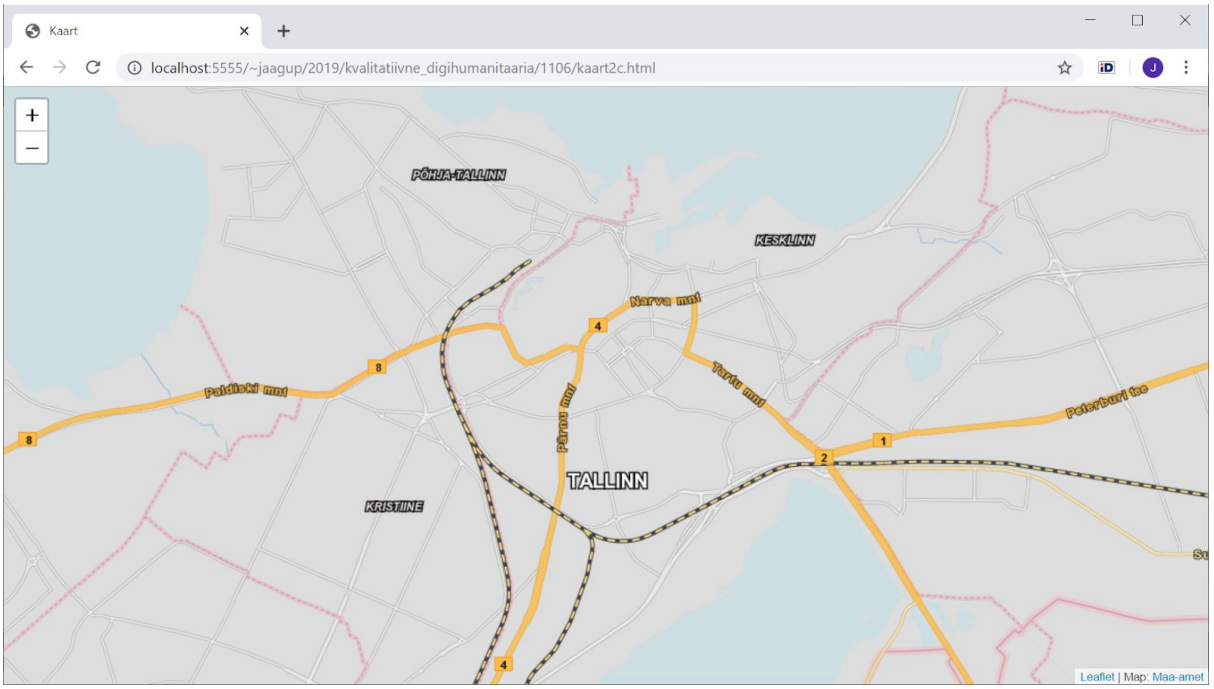
```
https://tiles.maaamet.ee/tm/tms/1.0.0/hybriid@GMC/11/1164/1448.png
&ASUTUS=TLU&ERIALA=DIGIHUMANITAARIA
```

kuvab näiteks enamiku Aegna saarest

Kaarti kuvav kood tervikuna

```
<!DOCTYPE html>
<html>
 <head>
 <title>Kaart</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.js"></script>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.css" />
 <style>
   body {padding: 0; margin: 0; }
   html, body { height: 100%; width: 100%;}
   img { border: 0;}
   #kaardikiht {float:bottom; width:100%; height:100%;}
 </style>
 <script>
   var kaart;
   function algus(){
    kaart = L.map('kaardikiht').setView([59.43, 24.75], 13);
    new L.TileLayer(
  'https://tiles.maaamet.ee/tm/tms/1.0.0/hybriid@GMC/{z}/{x}/{y}.png&ASUTUS=TLU&ERIALA=DIGIHUMANITAARIA', {
        attribution: "Map: <a  href='http://www.maaamet.ee/'>Maa-amet</a>",
        tms: true,
   }).addTo(kaart);
   }
  </script>
 </head>
 <body onload="algus()">
  <div id="kaardikiht"></div>
 </body>
</html>
```

ning nähtav tulemus.



Harjutus

- Pane näide tööle
- Kuva kaardil oma kodumaakond

OpenStreetMap

Sarnane kaart `mapbox.com`-i kaudu kättesaadava OpenStreetMap kaardipakkuja andmetega. Sealse võtme saamiseks tuleb end aga keskkonnas registreerida. Muu sama, vaid aadress teine pole vaja määrata, et kaarditükid tulevad TMS-projektsioonis. Kohustuslikult viisakas on näidata kaardiandmete pakkujat kaardil.

```
<!DOCTYPE html>
<html>
<head>

        <title>Kaart</title>

        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.js"></script>
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.css" />
        <script>
          var kaart;
          function algus(){
            kaart = L.map('kaardikiht').setView([59.43, 24.75], 13);
```

```
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=pk.eyJ1Ijoib
WFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {
                maxZoom: 18,
                id: 'mapbox.streets'
            }).addTo(kaart);
          }
        </script>
</head>
<body onload="algus()">
<div id="kaardikiht" style="width: 600px; height: 400px;"></div>

</body>
</html>
```

Hüpikmenüü

Leafletiga tuleb kaasa objekt nimega L, mil küljes mitmesuguseid käske. Käsklus popup() loob hüpikmenüü.

```
              hypik=L.popup();
              kaart.on("click", vajutus);
```

Kaardi peal hiireklõpsu peale käivitub funktsioon vajutus(), mis saab kaasa vajutuse andmed. Ühtlasi kuvatakse sealtkaudu laius- ning pikkuskraad.

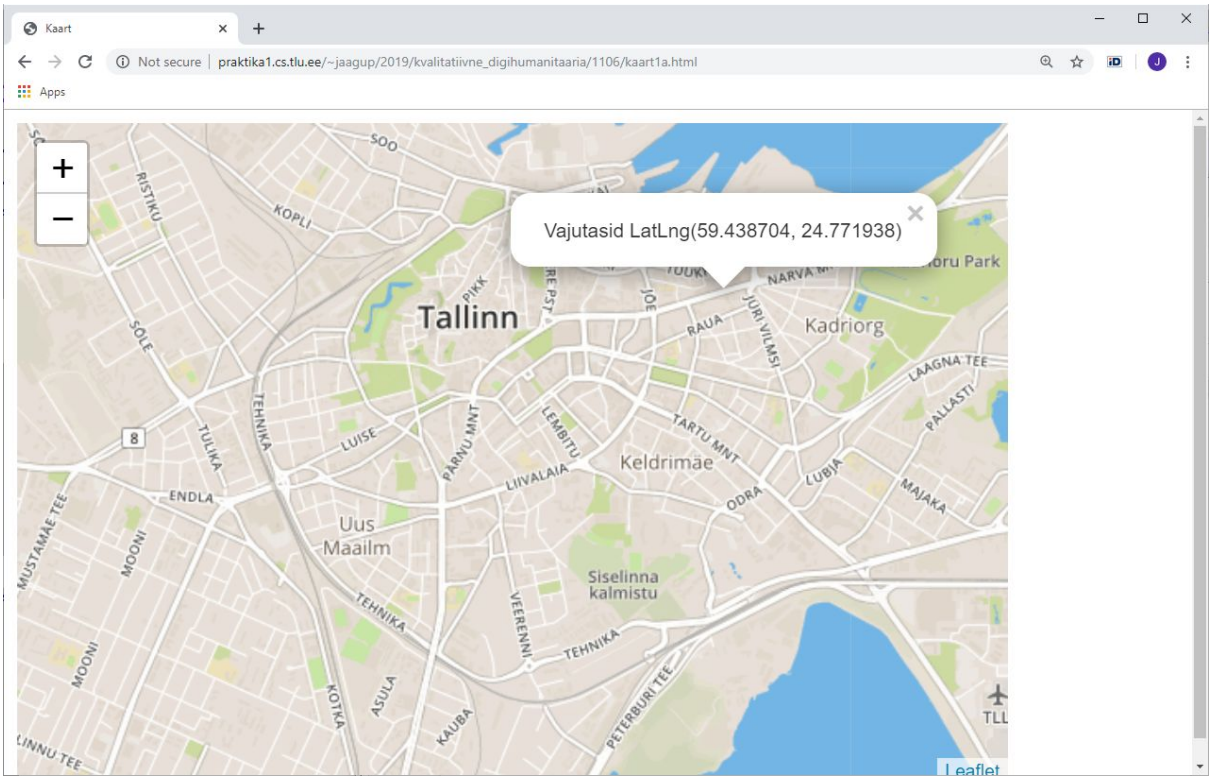
```
          function vajutus(e){
             hypik.setLatLng(e.latlng).
                 setContent("Vajutasid "+e.latlng.toString()).openOn(kaart);
          }
```

Kood tervikuna

```
<!DOCTYPE html>
<html>
<head>
        <title>Kaart</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.js"></script>
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.css" />
        <script>
          var kaart;
          var hypik;
          function algus(){
            kaart = L.map('kaardikiht').setView([59.43, 24.75], 13);

L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=pk.eyJ1Ijoib
WFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {
                maxZoom: 18,
                id: 'mapbox.streets'
            }).addTo(kaart);
              hypik=L.popup();
              kaart.on("click", vajutus);
```

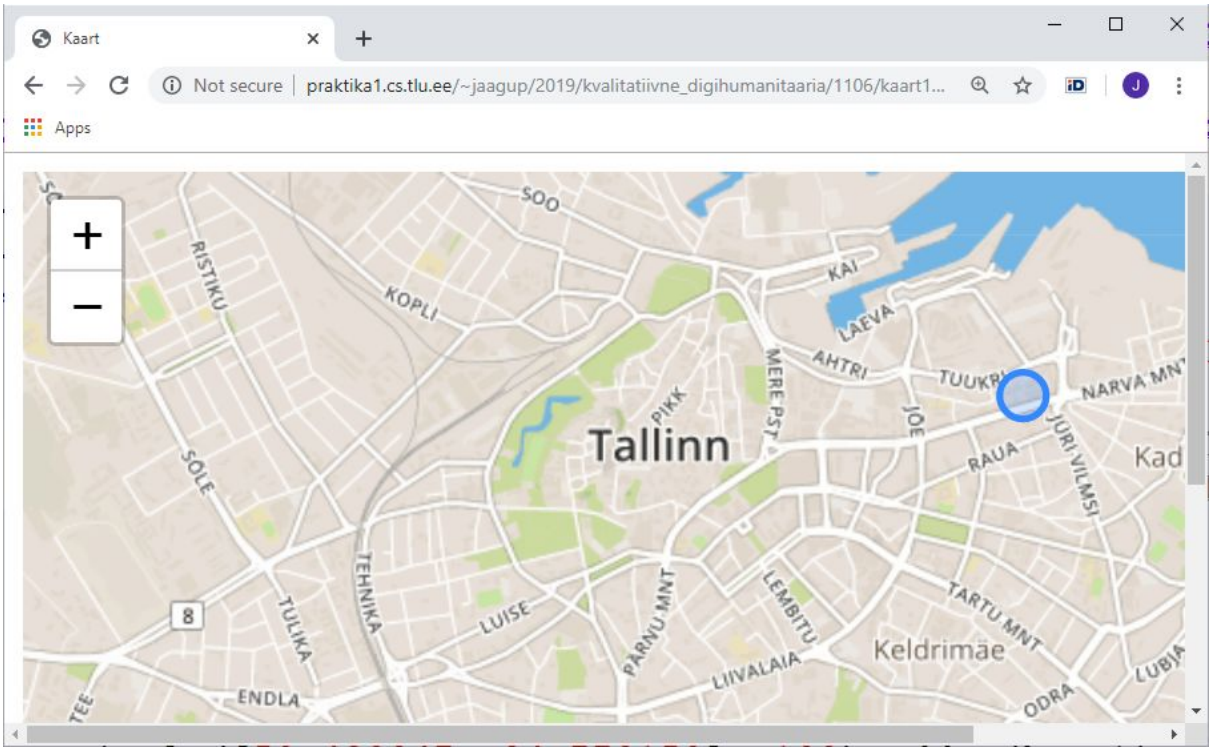
```
        }

        function vajutus(e){
           hypik.setLatLng(e.latlng).
              setContent("Vajutasid "+e.latlng.toString()).openOn(kaart);
        }
      </script>
</head>
<body onload="algus()">
<div id="kaardikiht" style="width: 600px; height: 400px;"></div>

</body>
</html>
```

Töö tulemus



Kaardile saab lisada markereid, ringe ja muud kraami (jooned, pinnad). Ringi lisamiseks sobib käsklus

```
L.circle([59.439047, 24.772159], 100).addTo(kaart);
```

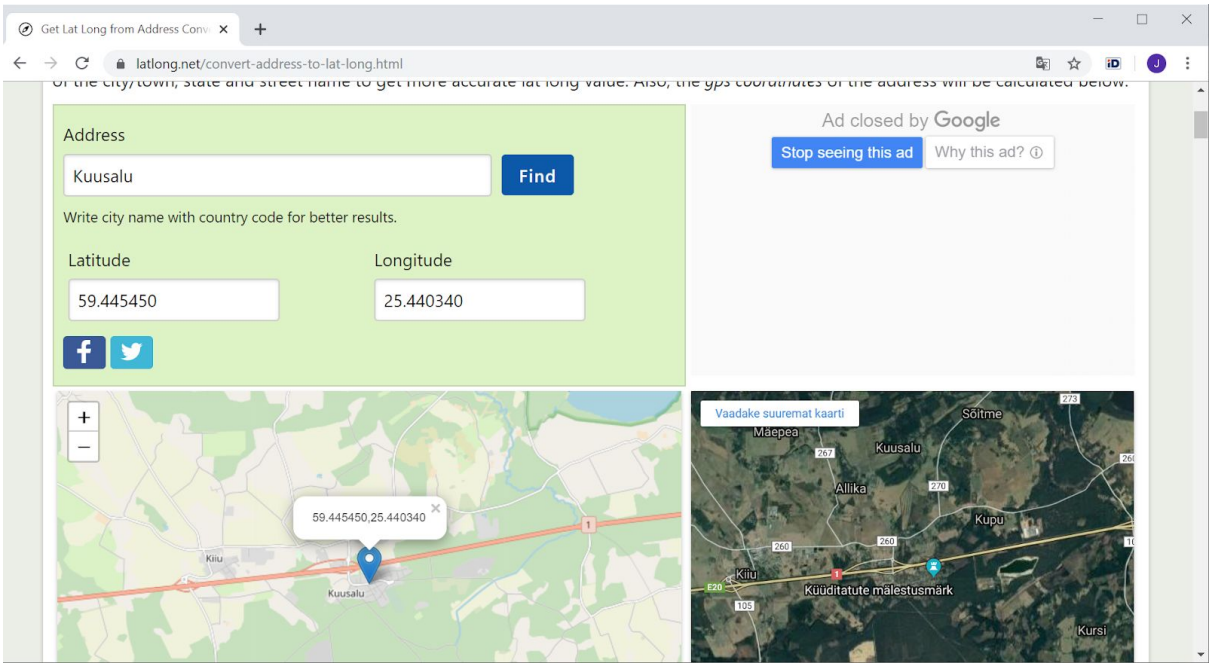


Harjutus

- Pane näide tööle
- Paiguta hüpikmenüü maa-ameti kaardile

Koordinaadid nimetuse järgi

Käsistsi ükshaaval asukohtade leidmiseks on mugav veebileht
`https://www.latlong.net/convert-address-to-lat-long.html`



Kui aadresse on vaja rohkem küsida, siis sobib teenus `api.opencagedata.com`. Selle pruukimiseks peab võtme registreerima ning võtmega saab päevas tasuta küsida 2500 vastet. Vastuse vorminguteks XML ning JSON.

`https://api.opencagedata.com/geocode/v1/json?q=Kuusalu&key=77ca3d44a80c40b6adac3efbaa8db31a&language=en&pretty=1`

Tuleb vastuseks päris pikk tekst, kust vaja siis sobivad väärtused üles otsida.


```
{
   "documentation" : "https://opencagedata.com/api",
   "licenses" : [
      {
         "name" : "see attribution guide",
         "url" : "https://opencagedata.com/credits"
      }
   ],
   "rate" : {
      "limit" : 2500,
      "remaining" : 2493,
      "reset" : 1573516800
   },
   "results" : [
      {
         "annotations" : {
            "DMS" : {
               "lat" : "59\u00b0 26' 42.63288'' N",
               "lng" : "25\u00b0 26' 3.80796'' E"
            },
            "MGRS" : "35VMF1122190668",
            "Maidenhead" : "KO29rk26du",
            "Mercator" : {
               "x" : 2831343.463,
               "y" : 8240405.148
            },
            "OSM" : {
               "edit_url" : "https://www.openstreetmap.org/edit?relation=355276#map=16/59.44518/25.43439",
               "url" : "https://www.openstreetmap.org/?mlat=59.44518&mlon=25.43439#map=16/59.44518/25.43439"
            },
```

```
      "UN_M49" : {
         "regions" : {
            "EE" : "233",
            "EUROPE" : "150",
            "NORTHERN_EUROPE" : "154",
            "WORLD" : "001"
         },
         "statistical_groupings" : [
            "MEDC"
         ]
      },
      "callingcode" : 372,
      "currency" : {
         "alternate_symbols" : [],
         "decimal_mark" : ",",
         "html_entity" : "&#x20AC;",
         "iso_code" : "EUR",
         "iso_numeric" : "978",
         "name" : "Euro",
         "smallest_denomination" : 1,
         "subunit" : "Cent",
         "subunit_to_unit" : 100,
         "symbol" : "\u20ac",
         "symbol_first" : 1,
         "thousands_separator" : "."
      },
      "flag" : "\ud83c\uddea\ud83c\uddea",
      "geohash" : "udd44v8v0rbw67sdc9s7",
      "qibla" : 158.61,
      "roadinfo" : {
         "drive_on" : "right",
         "speed_in" : "km/h"
      },
      "sun" : {
         "rise" : {
            "apparent" : 1573452180,
            "astronomical" : 1573443480,
            "civil" : 1573449420,
            "nautical" : 1573446360
         },
         "set" : {
            "apparent" : 1573480860,
            "astronomical" : 1573489560,
            "civil" : 1573483680,
            "nautical" : 1573486680
         }
      },
      "timezone" : {
         "name" : "Europe/Tallinn",
         "now_in_dst" : 0,
         "offset_sec" : 7200,
         "offset_string" : "+0200",
         "short_name" : "EET"
      },
      "what3words" : {
         "words" : "snaps.others.meddled"
      },
      "wikidata" : "Q2668988"
   },
   "bounds" : {
      "northeast" : {
         "lat" : 59.45018,
         "lng" : 25.455839
      },
      "southwest" : {
         "lat" : 59.4407034,
         "lng" : 25.4135192
      }
   },
   "components" : {
      "ISO_3166-1_alpha-2" : "EE",
```

```
            "ISO_3166-1_alpha-3" : "EST",
            "_type" : "neighbourhood",
            "city_district" : "Kuusalu alevik",
            "continent" : "Europe",
            "country" : "Estonia",
            "country_code" : "ee",
            "county" : "Kuusalu vald",
            "political_union" : "European Union"
         },
         "confidence" : 8,
         "formatted" : "Kuusalu alevik, Kuusalu vald, Estonia",
         "geometry" : {
            "lat" : 59.4451758,
            "lng" : 25.4343911
         }
      }
]}
```


Faili tervikkuju kättesaadav aadressil
`http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviis_asukoht_kuusalu.txt`

Pythonil on õnneks JSON-kujul tekstist sobivate andmete eraldamiseks sobiv teek olemas. Sisendfaili uurides paistab, et Kuusalu koordinaadid leiab massiivi `results` sektsioonist `geometry`.

```
import json
obj=json.loads(open("regiviis_asukoht_kuusalu.txt").read())
asukoht=obj["results"][0]["geometry"]
print(asukoht["lat"], asukoht["lng"])
```

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1104 $ python3.5
loe_json.py
59.4451758 25.4343911
```

Kõikide kihelkonnanimede leidmiseks ühendame viisid ja nende metaandmed tulpade FKey ja ID kaudu

```
import pandas as pd
viisid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt")
viisidmeta=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisidmeta.txt")
koos=viisid.merge(viisidmeta, left_on="FKey", right_on="ID")
print(list(sorted(set(filter(lambda t: isinstance(t,
(str)),koos["kihelkond"].values.tolist())))))
```

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1104 $ python3.5 asukohad3.py
['?', 'Ambla', 'Ambla>Jõhvi', 'Ambla>Kadrina', 'Ambla>Väike-Maarja', 'Anna', 'Anseküla', 'Antsla?', 'Audru',
'Audru>Järva-Madise', 'Emmaste', 'Haljala', 'Haljala>Kadrina', 'Haljala>Rakvere', 'Halliste', 'Hanila',
'Hanila>Varbla', 'Hargla', 'Hargla>Pärnu', 'Hargla>Rõuge', 'Harju-Jaani', 'Harju-Jaani [tänap. Jõelähtme v.]',
'Harju-Jaani>Ambla', 'Harju-Jaani>Kadrina', 'Harju-Jaani>Kuusalu', 'Harju-Jaani>Rakvere', 'Harjumaa', 'Helme',
'Helme? kust seegi info?', 'Häädemeeste', 'Häädemeeste k/n', 'Iisaku', 'Iisaku>Jõhvi', 'Jaani', 'Juuru',
'Juuru>Rapla', 'Jämaja', 'Järva-Jaani', 'Järva-Jaani [?Väika-Maarja]', 'Järva-Jaani>Ambla',
'Järva-Jaani>Narva', 'Järva-Jaani>Väike-Maarja', 'Järva-Madise', 'Järva-Madise>Ambla',
'Järva-Madise>Järva-Jaani?', 'Järvamaa', 'Jõelähtme', 'Jõhvi', 'Jõhvi>Iisaku', 'Jüri>Kuusalu', 'Kaarma',
'Kadria', 'Kadrina', 'Kadrina > Väike-Maarja', 'Kadrina>Ambla', 'Kadrina>Haljala', 'Kadrina>Rakvere',
'Kadrina>Väike-Maarja', 'Kadrina>Väikr-Maarja', 'Kambja', 'Kambja>Kanepi', 'Kanepi', 'Kanepi>Otepää', 'Karja',
'Karja>Vormsi', 'Karksi', 'Karksi>Halliste', 'Karula', 'Karula>Pilistvere', 'Karula>Suure-Jaani', 'Karuse',
'Karuse [Kullamaa?]', 'Karuse>Hanila', 'Karuse>Häädemeeste', 'Kaukasus', 'Keila', 'Kihelkonna',
'Kihelkonna>Anseküla', 'Kihnu', 'Kihnu?>Kodavere', 'Kingissepa', 'Kirbla', 'Kodavere', 'Kodavere>Jõhvi',
'Koeru', 'Koeru>Järva-Madise', 'Koeru>Jõhvi', 'Koeru>Laiuse', 'Koeru>Narva', 'Koeru>Peetri', 'Koeru>Simuna',
```

```
'Koeru>Väike-Maarja', 'Koeru?/Järva-Madise?', 'Koeru?>Ambla', 'Kolga-Jaani', 'Kolga-Jaani>Kodavere',
'Kolga-Jaani>Laiuse', 'Kolga-Jaani>Rannu', 'Kose', 'Kose>Harju-Jaani', 'Kose>Järva-Madise', 'Kraasna', 'Krimm',
'Kullamaa', 'Kullamaa>Haapsalu', 'Kursi', 'Kuusalu', 'Kuusalu>Jamburi-Simititsa', 'Kuusalu>Mustamere r.',
'Käina', 'Kärla', 'Kõpu', 'Kõrgepalu', 'Laiuse', 'Laiuse<Torma?', 'Laiuse>Maarja-Magdaleena', 'Laiuse>Torma',
'Leivu', 'Lihula', 'Lihula>Kullamaa', 'Lihula>Rapla', 'Lutsi', 'Lääne-Nigula', 'Läänemaa', 'Läänemaa (ranna
Läänemaa)', 'Lüganuse', 'Lüganuse>Jõhvi', 'Maarja-Magdaleena', 'Martna', 'Martna>Rapla', 'Mihkli', 'Muhu',
'Muhu>Karuse', 'Muhu?', 'Mustjala', 'Mustjala>Anseküla', 'Mustjala>Kärla', 'Märjamaa', 'Märjamaa>Hageri',
'Nissi', 'Noarootsi', 'Noarootsi>Lääne-Nigula', 'Nõo', 'Nõo>Tartu-Maarja?', 'Otepää', 'Otepää>Haapsalu',
'Otepää>Mihkli', 'Paistu', 'Paistu>Halliste', 'Paistu>Helme', 'Paistu>Koeru', 'Paistu>Suure-Jaani',
'Paistu>Türi', 'Palamuse', 'Peetri', 'Peetri>Järva-Jaani', 'Peetri>Rakvere', 'Peetri>Torma', 'Peterburg',
'Petserimaa', 'Pilistvere', 'Pilistvere>Ambla', 'Pilistvere>Peterburg', 'Pilistvere>Rakvere', 'Pilisvere',
'Puhja', 'Puhja>Nõo', 'Puhja>Tartu', 'Pärnu', 'Pärnu-Jaagupi', 'Pärnumaa', 'Põhja-Kaukaasia eesti asundus',
'Põltsamaa', 'Põltsamaa>Kursi', 'Põltsamaa>Lüganuse>Jõhvi', 'Põlva', 'Põlva>Räpina', 'Pöide',
'Püha>Harju-Jaani', 'Pühalepa', 'Rakvere', 'Rannu', 'Rapla', 'Reigi', 'Ridala', 'Ridala>Haapsalu', 'Risti',
'Ruhnu', 'Räpina', 'Rõngu', 'Rõuge', 'Rõuge>Setu', 'Rõuge?', 'Rõuge?=Vastseliina?', 'Saarde', 'Sangaste',
'Sangaste>Otepää', 'Setu', 'Setu, Järvesuu v.', 'Setu, Meremäe v.', 'Setu, Meremäe v.>Misso', 'Setu, Misso v.',
'Setu, Misso v.>Kambja', 'Setu, Misso v.>Nõo al.', 'Setu, Mäe v.', 'Setu, Petseri v.', 'Setu, Saatse v.',
'Setu, Saatse v.>MMg', 'Setu, Saatse v.>Sangaste', 'Setu, Vilo v.', 'Setu, Vilo v.>Räpina', 'Setu>Karula',
'Setu>Rõuge', 'Simuna', 'Simuna>Haljala', 'Simuna>Jõhvi', 'Suure-Jaani', 'Suure-Jaani>Järva-Madise',
'Suure-Jaani>Türi', 'Tallinn', 'Tallinn>Laiuse-Tähtvere', 'Tallinn>Orarino', 'Tartu', 'Tartu l.',
'Tartu-Maarja', 'Tartu/Paistu', 'Tartumaa', 'Tarvastu', 'Tarvastu>Kambja', 'Tarvastu>Puhja', 'Tarvastu>Rõuge',
'Tarvastu>Tallinn', 'Tarvastu>Türi', 'Tori', 'Tori>Suure-Jaani', 'Torma', 'Tõstamaa', 'Tõstamaa/Audru',
'Tõstamaa? Hanila?', 'Tõstamaa? Hanila? Mihkli?', 'Türi', 'Türi>Ambla', 'Türi>Jõhvi', 'Türi>Peetri',
'Türi>Pilistvere', 'Türi>Rapla', 'Urvaste', 'Urvaste(<Karula?)', 'Urvaste>Karula', 'Vaivara', 'Vaivara>Jõhvi',
'Vana-Antsla v.', 'Varbla', 'Varbla>Tõstamaa', 'Vastseliina', 'Vastseliina?=Rõuge?', 'Vigala', 'Vigala/Karuse',
'Vigala>Kullamaa', 'Vigala>Märjamaa', 'Viljandi', 'Viljandi>Pillistvere', 'Viljandi?>Tartu', 'Viru-Jaagupi',
'Viru-Jaagupi>Väike-Maarja', 'Viru-Nigula', 'Viru-Nigula>Haljala', 'Viru-Nigula>Rakvere', 'Vormsi',
'Väike-Maarja', 'Väike-Maarja>Ambla', 'Väike-Maarja>Kadrina', 'Vändra', 'Vändra>Kose', 'Vändra>Pärnu-Jaagupi',
'Vändra?', 'Võnnu', 'Võru v.>Vastseliina', 'Võrumaa', 'karksi', 'Äksi', 'Äksi>Jõhvi']
```

Veebiaadressis kasutamiseks tuleb erisümbolid asendada nende jaoks vajalike tähekombinatsioonidega. Seda mõistab käsklus urllib.parse.quote

```
>>> import urllib.parse
>>> urllib.parse.quote("Koeru>Väike-Maarja")
'Koeru%3EV%C3%A4ike-Maarja'
```

Kahe näite koostöös valmib fail, kus iga eri asukoha tarbeks küsitakse selle koordinaadid opencagedata.com lehelt ning salvestatakse tulemus faili

```
import urllib.parse
import urllib.request
import json
import pandas as pd
viisid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt")
viisidmeta=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisidmeta.txt")
koos=viisid.merge(viisidmeta, left_on="FKey", right_on="ID")
paiganimed=list(sorted(set(filter(lambda t: isinstance(t,
(str)),koos["kihelkond"].values.tolist()))))

f=open("koordinaadid.txt", "w")
for paiganimi in paiganimed:
 try:
  print(paiganimi)

veebiaadress="https://api.opencagedata.com/geocode/v1/json?q="+urllib.parse.quote(paiganimi
)+"&key=77ca3d44a80c40b6adac3efbaa8db31a&language=en&pretty=1"
  veebisisu=urllib.request.urlopen(veebiaadress).read().decode("utf-8")
  obj=json.loads(veebisisu)
  asukoht=obj["results"][0]["geometry"]
  f.write(paiganimi+","+str(asukoht["lat"])+","+str(asukoht["lng"])+"\n")
 except:
  print("vigane ", paiganimi)
```

```
f.close()
```

Käivitus

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1104 $ python3.5
asukohad3.py
?
vigane  ?
Ambla
Ambla>Jõhvi
```

...

ja valminud fail

```
Ambla,59.1776682,25.7162154
Ambla>Jõhvi,59.35917,27.42111
Ambla>Kadrina,59.33472,26.145
Ambla>Väike-Maarja,59.12639,26.25
Anna,10.416667,77.666667
Anseküla,58.096624,22.2180925
Antsla?,57.8754879,26.491877
Audru,58.4094616,24.3438522
Audru>Järva-Madise,59.11639,25.65861
Emmaste,58.7021644,22.5801435
```

tervikuna nähtav

```
http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid_koordinaadid.txt
```

Kontroll lugemiseks, et kas saadakse andmed kätte. Vastatakse, et real 178 on sellega probleem.

```
>>> import pandas as pd
>>> kohad=pd.read_csv("koordinaadid.txt")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.5/dist-packages/pandas/io/parsers.py", line 655, in parser_f
pandas.errors.ParserError: Error tokenizing data. C error: Expected 3 fields in line 178,
saw 4

Setu, Vilo v.>Räpina,58.09806,27.46361
```

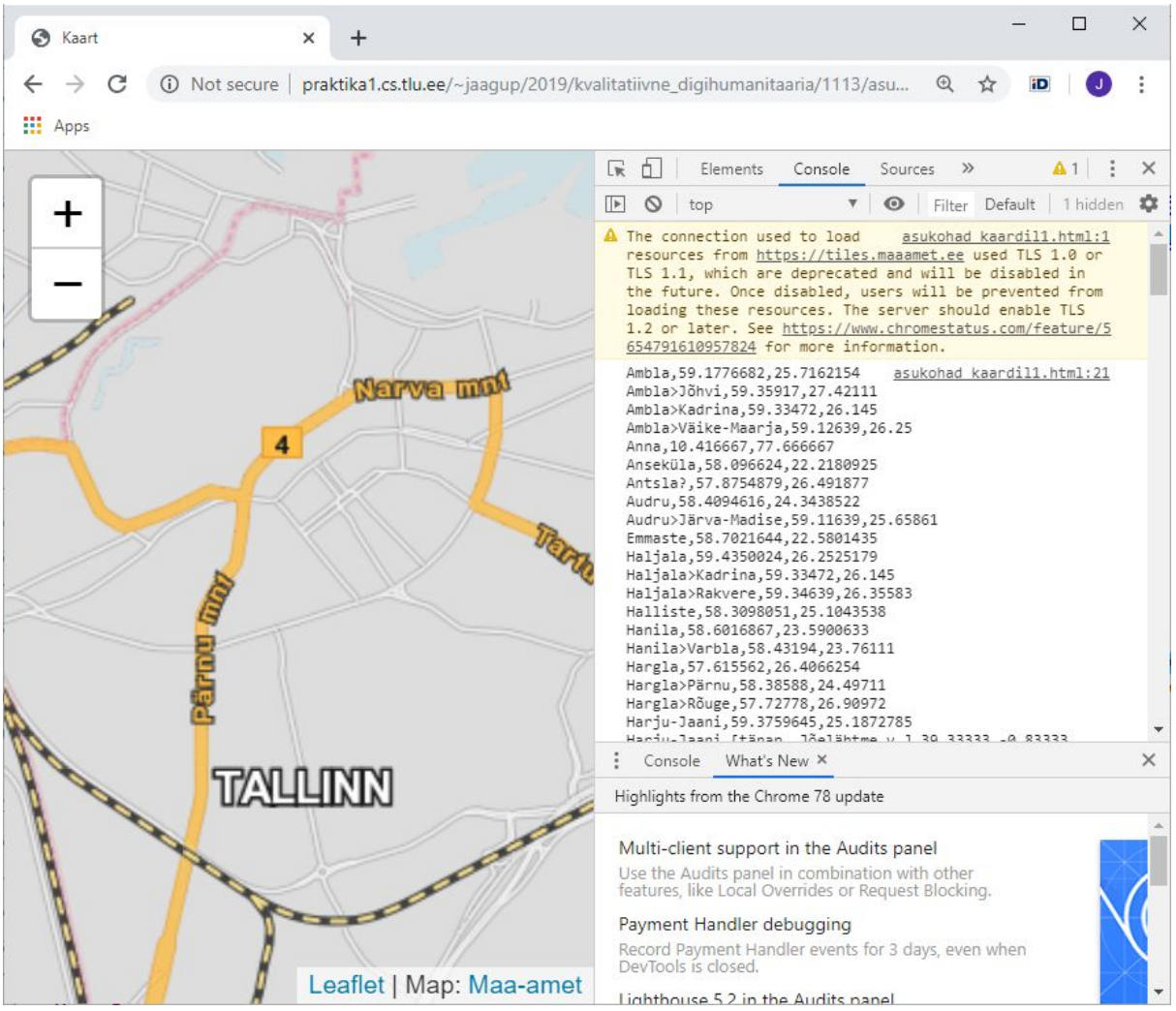
Lähemal uurimisel paistab, et tegemist reaga, kus ka aadressis eneses on sees koma - nii ei saa csv-lugeja aru, kus on tulpade vahe. Lihtsama lahendusena eemaldame koma aadressist. Kui sarnaseid aadresse oleks palju, siis tuleks aga miski süstemaatilisem lahendus välja mõelda - olgu kõikide aadressisiseste komade eemaldamine või aadressile jutumärkide ümber panek

```
>>> kohad=pd.read_csv("koordinaadid.txt")


Setu Vilo v.>Räpina,58.09806,27.46361
```

Koordinaatide kuvamine kaardile

Alustuseks näide, kus kuvatakse kaart ning loetakse serverist kihelkondade koordinaadid

```
<!DOCTYPE html>
<html>
 <head>
 <title>Kaart</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.js"></script>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.css" />
 <style>
   body {padding: 0; margin: 0; }
   html, body { height: 100%; width: 100%;}
   img { border: 0;}
   #kaardikiht {float:bottom; width:100%; height:100%;}
 </style>
 <script>
   var kaart;
   var xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;

   function loeAndmed(){
     if(xhr.readyState==4){ //kui andmed kohal
      console.log(xhr.responseText);
        }
   }
   function algus(){
    xhr.open("GET", "koordinaadid.txt", true);
       xhr.send();
    kaart = L.map('kaardikiht').setView([59.43, 24.75], 13);
    new
L.TileLayer('https://tiles.maaamet.ee/tm/tms/1.0.0/hybriid@GMC/{z}/{x}/{y}.png&ASUTUS=TLU&E
RIALA=DIGIHUMANITAARIA', {
       attribution: "Map: <a  href='http://www.maaamet.ee/'>Maa-amet</a>",
       tms: true,
   }).addTo(kaart);
   }
  </script>
 </head>
 <body onload="algus()">
  <div id="kaardikiht"></div>
 </body>
</html>
```



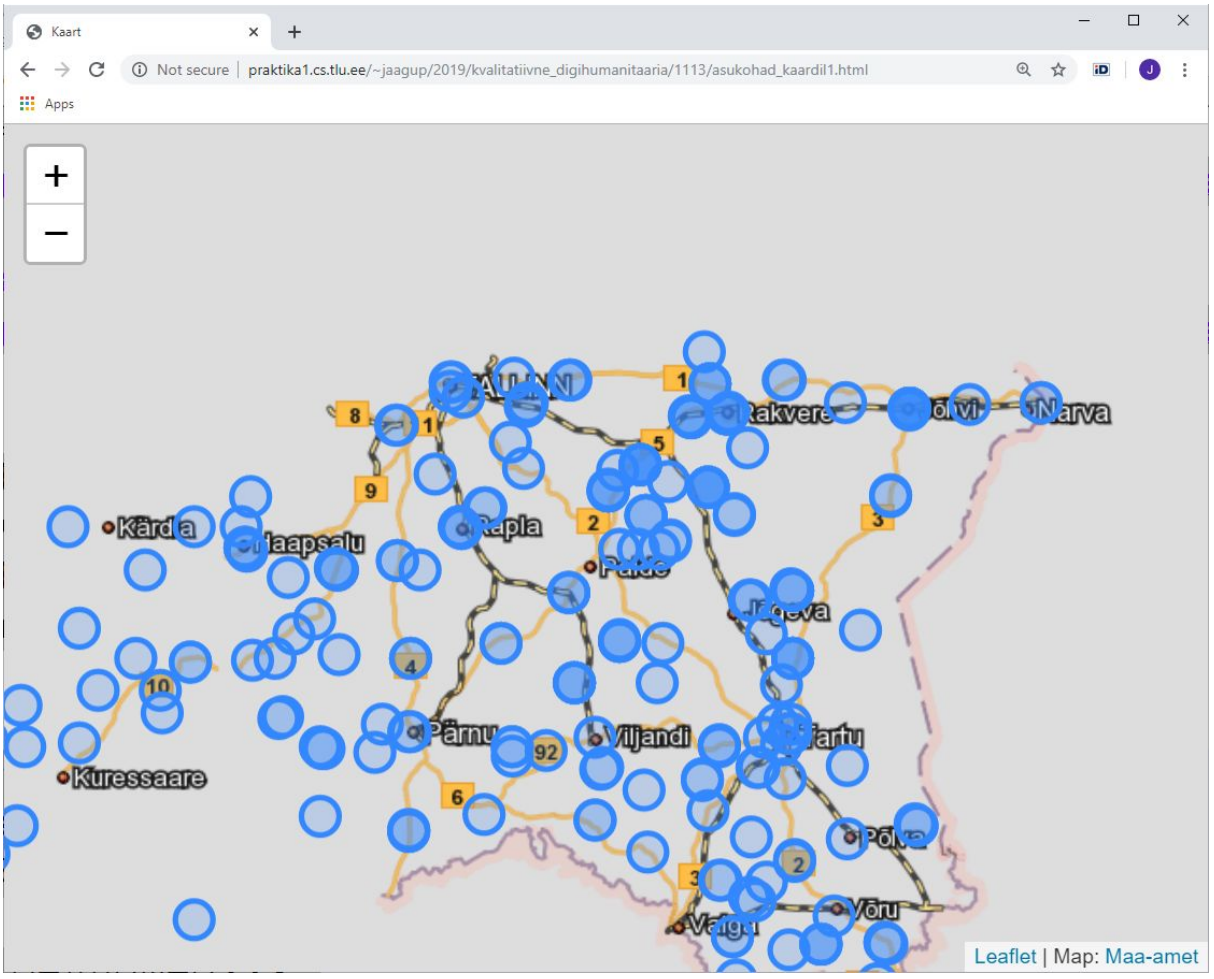
Leht toimivana aadressil

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/kaart_asukohad1.html`

Juurde osa, kus iga koha marker paigutatakse kaardile

```
<!DOCTYPE html>
<html>
 <head>
 <title>Kaart</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.js"></script>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.css" />
 <style>
   body {padding: 0; margin: 0; }
   html, body { height: 100%; width: 100%;}
   img { border: 0;}
   #kaardikiht {float:bottom; width:100%; height:100%;}
 </style>
 <script>
   var kaart;
   var xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
```

```
   function loeAndmed(){
     if(xhr.readyState==4){ //kui andmed kohal
      //console.log(xhr.responseText);
          var m=xhr.responseText.split("\n");
          for(var i=0; i<m.length; i++){
             var rida=m[i].split(",");
                L.circleMarker(
                 [parseFloat(rida[1]), parseFloat(rida[2])]).
                 addTo(kaart);
          }
         }
   }
   function algus(){
    xhr.open("GET", "koordinaadid.txt", true);
        xhr.send();
    kaart = L.map('kaardikiht').setView([59.43, 24.75], 13);
    new
L.TileLayer('https://tiles.maaamet.ee/tm/tms/1.0.0/hybriid@GMC/{z}/{x}/{y}.png&ASUTUS=TLU&E
RIALA=DIGIHUMANITAARIA', {
        attribution: "Map: <a  href='http://www.maaamet.ee/'>Maa-amet</a>",
        tms: true,
   }).addTo(kaart);
   }
  </script>
 </head>
 <body onload="algus()">
  <div id="kaardikiht"></div>
 </body>
</html>
```



Tulemus veebilehena

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/kaart_asukohad2.html`

Harjutus

- Pane näide tööle
- Jäta tekstifaili vaid kodumaakonna paari kihelkonna koordinaadid. Kuva kaart ekraanile koos nende kohta käivate markeritega

Valitud algusnootidega viiside asukohad

Siiani oli koordinaatide failis vaid kihelkonna või küla asukoht. Nüüd aga püüame iga viisi juurde märkida koordinaadid, kust see üles kirjutatud. Loome iga algusnoodi kohta eri faili. Siin näites h-noodiga algavad viisid ja nende asukohad.

```
import pandas as pd
viisid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt")
viisidmeta=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisidmeta.txt")
```

```
koordinaadid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid_koordinaa
did.txt", header=None).rename(columns={0:"kohanimi",1:"laiuskraad",2:"pikkuskraad"})
koos=viisid.merge(viisidmeta, left_on="FKey", right_on="ID")
koos=koos.merge(koordinaadid, left_on="kihelkond", right_on="kohanimi")
koos[koos["P1"]=="h"][["kohanimi", "laiuskraad", "pikkuskraad"]].to_csv("algush.txt",
index=False)
```

Tulemus järgmine:

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1113 $ more algush.txt
kohanimi,laiuskraad,pikkuskraad
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
Ambla,59.17766820000001,25.7162154
```

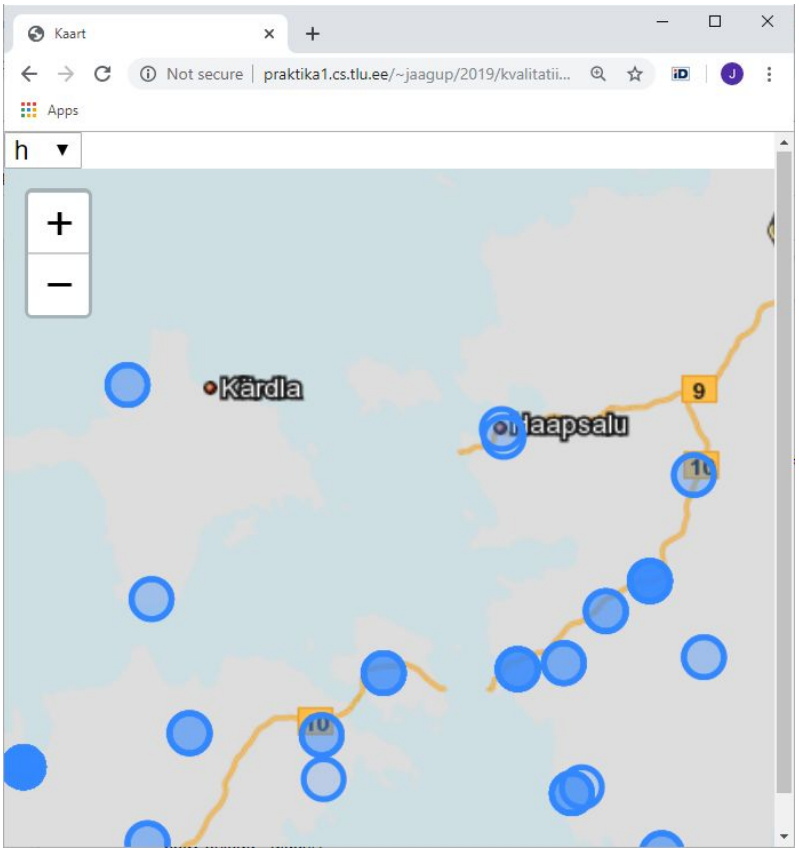
ehk siis sama koha peale tuleb mitu viisi.

Kuvamine:

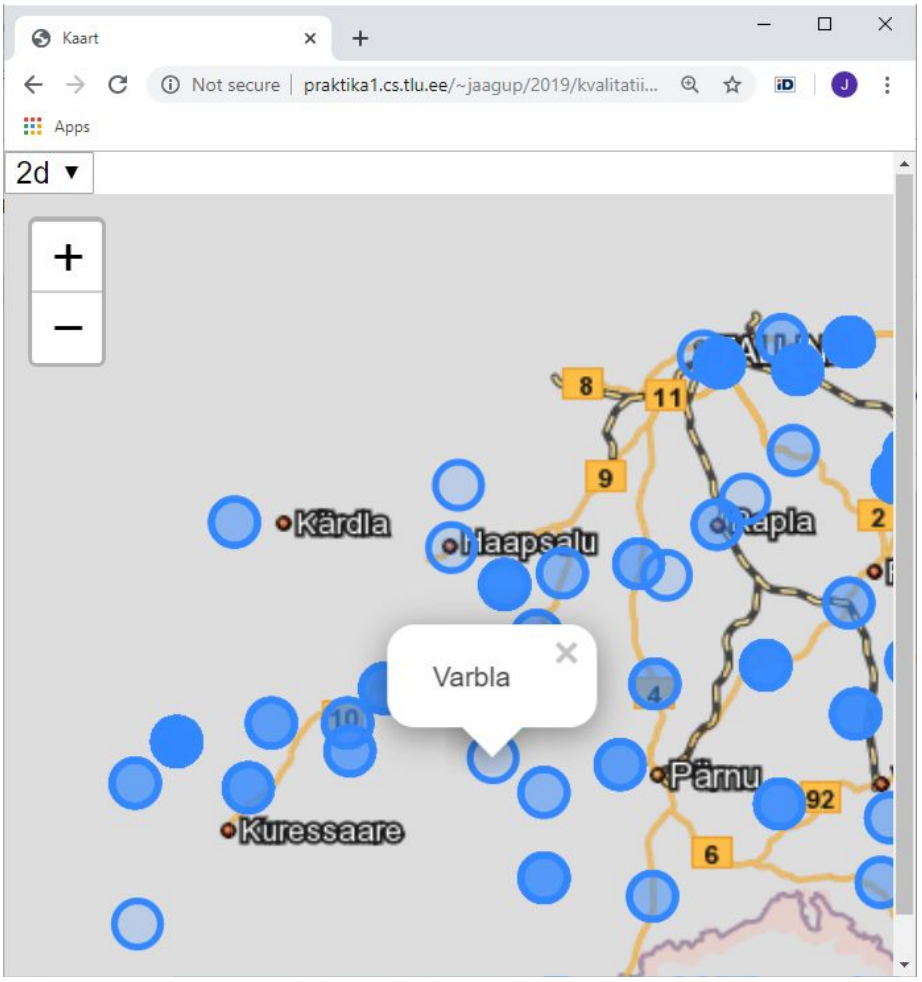
```
<!DOCTYPE html>
<html>
 <head>
 <title>Kaart</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.js"></script>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.5.1/leaflet.css" />
 <style>
   body {padding: 0; margin: 0; }
   html, body { height: 100%; width: 100%;}
   img { border: 0;}
   #kaardikiht {float:bottom; width:85%; height:85%;}
 </style>
 <script>
   var kaart, grupp1;
   var xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;

   function loeAndmed(){
     if(xhr.readyState==4){
         grupp1.clearLayers();
         var m=xhr.responseText.split("\n");
         for(var i=0; i<m.length; i++){
            var rida=m[i].split(",");
                try{
                grupp1.addLayer(L.circleMarker(
                 [parseFloat(rida[1]), parseFloat(rida[2])]).
                 addTo(kaart).bindPopup(rida[0]));
                } catch(viga){console.log(viga)}
         }
         console.log(grupp1);
        }
   }
```

```
   function algus(){
    kaart = L.map('kaardikiht').setView([58.6, 25], 7);
    new
L.TileLayer('https://tiles.maaamet.ee/tm/tms/1.0.0/hybriid@GMC/{z}/{x}/{y}.png&ASUTUS=TLU&E
RIALA=DIGIHUMANITAARIA', {
        attribution: "Map: <a  href='http://www.maaamet.ee/'>Maa-amet</a>",
        tms: true,
   }).addTo(kaart);
   grupp1=L.featureGroup().addTo(kaart);
    xhr.open("GET", "algus2d.txt", true);
        xhr.send();
   }

   function valikuMuutus(){
      var fnimi="algus"+
           document.getElementById("valik1").value+".txt";
      xhr.open("GET", fnimi, true);
         xhr.send();
   }
  </script>
 </head>
 <body onload="algus()">
  <select id="valik1" onchange="valikuMuutus()">
    <option>2d</option>
    <option>h</option>
    <option>ggah</option>
  </select>
  <div id="kaardikiht"></div>
 </body>
</html>
```

Viisid, mille esimeseks noodiks on h:



Viisid, mille algusnoodiks on 2d:



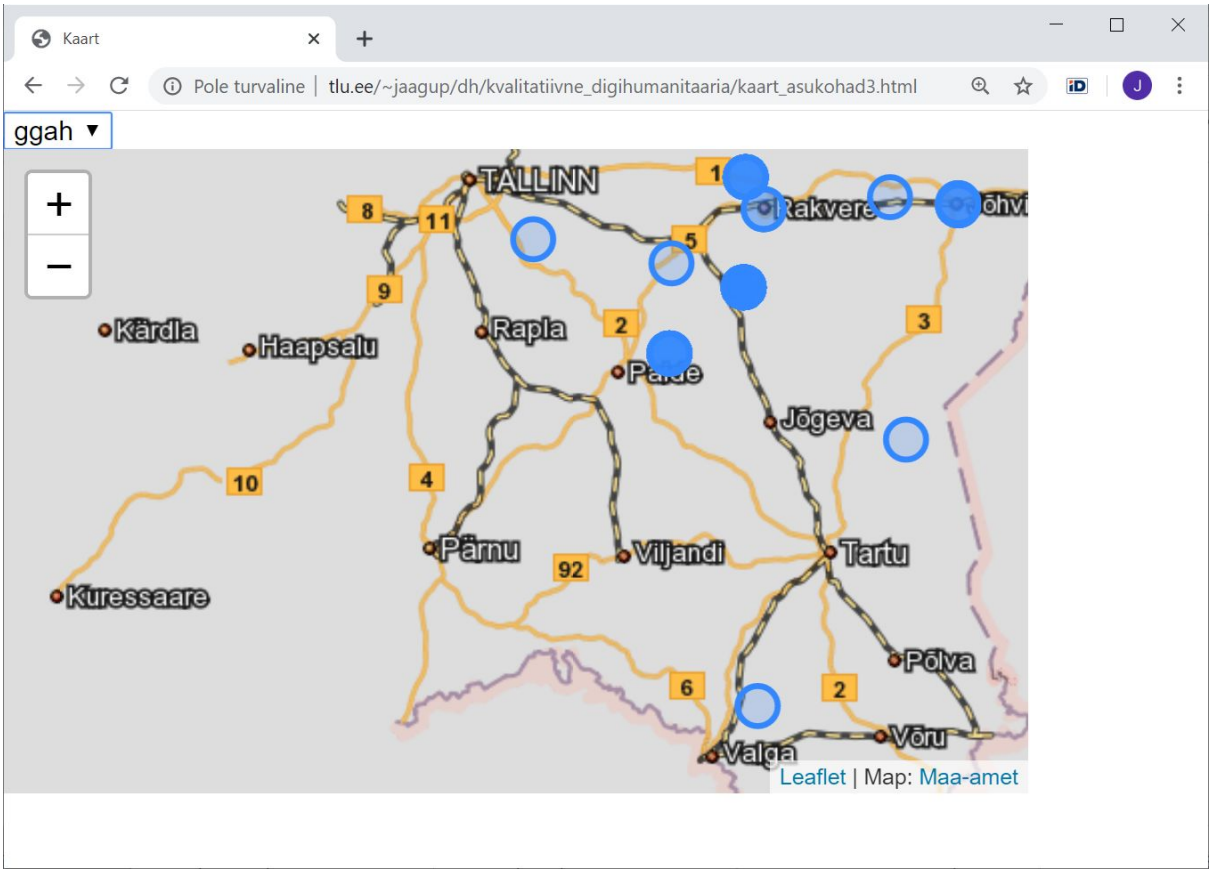
Pikema alguslõigu järgi filtreerimiseks tuleb lihtsalt mitut nooti võrrelda.

```
import pandas as pd
viisid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid.txt")
viisidmeta=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisidmeta.txt")
koordinaadid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/muu/muusika/regiviisid_koordinaa
did.txt", header=None).rename(columns={0:"kohanimi",1:"laiuskraad",2:"pikkuskraad"})
koos=viisid.merge(viisidmeta, left_on="FKey", right_on="ID")
koos=koos.merge(koordinaadid, left_on="kihelkond", right_on="kohanimi")
algus=["g", "g", "a", "h"]
koos[(koos["P1"]==algus[0]) & (koos["P2"]==algus[1]) & (koos["P3"]==algus[2]) &
(koos["P4"]==algus[3])][["kohanimi", "laiuskraad", "pikkuskraad"]].to_csv("algusggah.txt",
index=False)
```

Nagu failisuuruste järgi näha, siis tuli fail märgatavalt lühem:

```
[jaagup@lin2 kvalitatiivne_digihumanitaaria]$ ls -la algus*
-rw-rw-r--  1 jaagup jaagup 31435 Nov 13 15:10 algus2d.txt
-rw-rw-r--  1 jaagup jaagup  3128 Nov 13 15:43 algusggah.txt
-rw-rw-r--  1 jaagup jaagup 24886 Nov 13 15:42 algush.txt
```

Joonise järgi näha, et `ggah`-algusega viisid on üles kirjutatud peamiselt idapoolsest Eestist, mõni üksik heledam ring leida ka mujalt.



Harjutus

- Pane näited tööle
- Lisa loetellu g-algusega viisid
- Leia g-tugiheliga viiside hulgast kümme leivinuimat algusnelikut. Koosta kaart, kus saab valida neliku ning näha nende viiside paiknemisi.

MIDI helid

Kuivõrd muusika ja kunstid kuuluvad humanitaarvaldkonna alla, selle üle vahel vaieldakse. Mõningane seos neil omavahel igatahes on - näiteks lauludes saavad kokku nii teksti- kui viisipool.

Digivaldkonnas vanaks ja lihtsaks helilõikude esitamise mooduseks on MIDI - Musical Instrument Digital Interface. Lihtsamal kujul antakse teada, millisel kõrgusel ja millal noot kõlab. Süntesaatori - olgu siis riist- või tarkvaralise - ülesandeks on valitud pilliga vastav noot kõlama panna.

Veebis on MIDI-häälte kõlama panekuks levinud vahend nimega Midi.js.

`https://github.com/mudcube/MIDI.js/`

Laeme alla

Pakime lahti

```
jaagup@praktika1 ~/public_html/2019/kvalitatiivne_digihumanitaaria/1118 $ unzip MIDI.js-master.zip
```

Käivitamiseks peab see olema veebiserveris, kohalikust kataloogist ei saa brauser vajalikke faile kätte. Kataloogis `examples` leiab paar töötavat näidet. Neist saab mõne aluseks võtta, kus näha, et millised failid vaja käivitamiseks sisse lugeda. Tekib objekt nimega MIDI, millele tuleb siis pärast lehe laadimist öelda:

```
MIDI.loadPlugin({soundfontUrl: "./soundfont/",
    instrument: "acoustic_grand_piano"});
```

Klaver on kohe olemas, teisi instrumente peab hiljem vajadusel juurde laadima.

Lihtsam näide heli tekitamise kohta. Iga nupuvajutuse kohta kõlab C-noot, ehk noot helikõrgusega 60 MIDI-süsteemis.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script src="../inc/shim/Base64.js" type="text/javascript"></script>
        <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
        <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
        <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
        <script src="../js/midi/gm.js" type="text/javascript"></script>
        <script src="../js/midi/loader.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
        <script>
          function algus(){
            MIDI.loadPlugin({
                soundfontUrl: "./soundfont/",
                instrument: "acoustic_grand_piano"});
          }
          function vajutus(){
            MIDI.noteOn(0, 60, 127);
          }

          function lahti(){
            MIDI.noteOff(0, 60);
          }
        </script>
</head>
<body onload="algus()">
  <input type="button" value="C"
```

```
    onmousedown='vajutus()' onmouseup="lahti()" />
</body>
</html>
```

Harjutus

- Koostage leht viie nupuga (C D E F G), pange igaühe vajutamisel helisema vastava MIDI-kõrgusega noot (60 62 64 65 67). Mängige nt. Rongisõitu nende nuppude peal

Lahendus:

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
       <script src="../inc/shim/Base64.js" type="text/javascript"></script>
       <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
       <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
       <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
       <script src="../js/midi/gm.js" type="text/javascript"></script>
       <script src="../js/midi/loader.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
       <script>
         function algus(){
           MIDI.loadPlugin({
               soundfontUrl: "./soundfont/",
               instrument: "acoustic_grand_piano"});
         }
         let helikorgused={
           "C": 60,
               "D": 62,
               "E": 64,
               "F": 65,
               "G": 67
         };
         function vajutus(nupp){
           MIDI.noteOn(0, helikorgused[nupp.value], 127);
         }

         function lahti(nupp){
           MIDI.noteOff(0, helikorgused[nupp.value]);
         }
       </script>
</head>
<body onload="algus()">
  <input type="button" value="C" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="D" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="E" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="F" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="G" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
</body>
</html>
```

Väljanägemine

ja kasutusaadress

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/noot2.html`

Valitud helistiku noodid

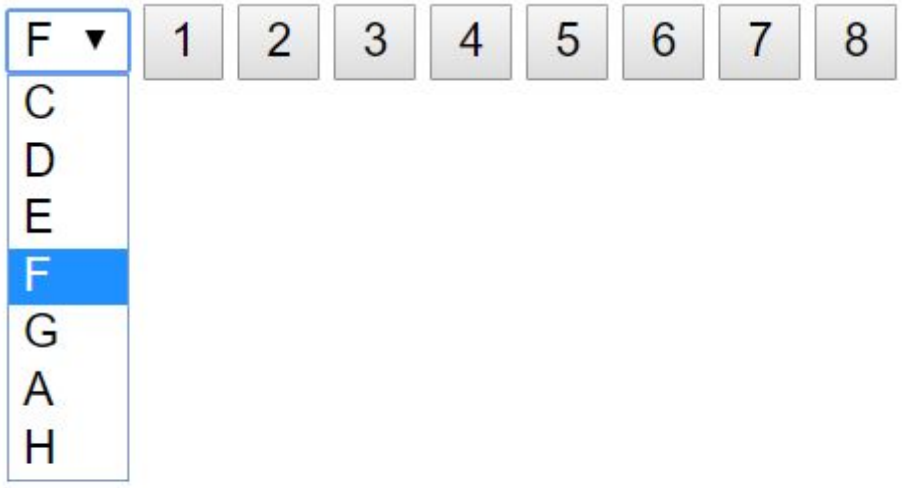
Täiendus lehele. Rippmenüüst saab valida, et millise helistikuga on tegu ning nupu peale vajutades võetakse sellest helistikust niimitmes aste.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script src="../inc/shim/Base64.js" type="text/javascript"></script>
        <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
        <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
        <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
        <script src="../js/midi/gm.js" type="text/javascript"></script>
        <script src="../js/midi/loader.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
        <script>
          function algus(){
            MIDI.loadPlugin({
                soundfontUrl: "./soundfont/",
                instrument: "acoustic_grand_piano"});
          }
          let helikorgused={
            "C": 60,
                "D": 62,
                "E": 64,
                "F": 65,
                "G": 67,
                "A": 69,
                "H": 71
          };
          let astmed={
            1:0, 2:2, 3:4, 4:5, 5:7, 6:9, 7:11, 8:12,
                 9:14, 10:16, 11:17, 12:19, 13:21, 14:23, 15:24
          }
          function korgus(nupp){
             return helikorgused[document.getElementById("helistikuvalik").value]+
                        astmed[nupp.value];
          }
          function vajutus(nupp){
            MIDI.noteOn(0, korgus(nupp), 127);
          }
```

```
          function lahti(nupp){
            MIDI.noteOff(0, korgus(nupp));
          }
        </script>
</head>
<body onload="algus()">
  <select id="helistikuvalik">
     <option>C</option>
     <option>D</option>
     <option>E</option>
     <option>F</option>
     <option>G</option>
     <option>A</option>
     <option>H</option>
  </select>
  <input type="button" value="1" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="2" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="3" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="4" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="5" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="6" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="7" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
  <input type="button" value="8" onmousedown='vajutus(this)' onmouseup="lahti(this)" />
</body>
</html>
```

Proovimiskoht

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/noot3.html`



Järjestikku kõlavad noodid

Mitme noodi järjest kõlama panekuks võib `MIDI.noteOn` käsule viimaseks parameetriks anda ooteaja sekundites. C-duuri üksikute nootidega kolmkõla tarbeks praegusel juhul C ehk

60 hakkab kõlama kohe vajutuse peale. E ehk 64 poole sekundi pärast ning G ehk 67 terve sekundi pärast.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script src="../inc/shim/Base64.js" type="text/javascript"></script>
        <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
        <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
        <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
        <script src="../js/midi/gm.js" type="text/javascript"></script>
        <script src="../js/midi/loader.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
        <script>
          function algus(){
            MIDI.loadPlugin({
                soundfontUrl: "./soundfont/",
                instrument: "acoustic_grand_piano"});
          }
          function vajutus(){
            MIDI.noteOn(0, 60, 127, 0);
            MIDI.noteOn(0, 64, 127, 0.5);
            MIDI.noteOn(0, 67, 127, 1);
          }
        </script>
</head>
<body onload="algus()">
  <input type="button" value="duurkolmkõla" onmousedown='vajutus()'  />
</body>
</html>
```

Veebiaadress:
`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/noot4.html`

Harjutus

- Lisage nupp mollkolmkõla tarbeks. Keskmine noot on poole tooni ehk ühe MIDI numbri võrra madalam
- Pange eelmisest näitest tagasi helistiku valik. Algusnoot, mis siin näites on 60, tuleb vastavalt rippmenüüst valitud helistiku tähele. C-60, D-62, E-64. Muud noodid ka samavõrra kõrgemad

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script src="../inc/shim/Base64.js" type="text/javascript"></script>
        <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
        <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
        <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
        <script src="../js/midi/gm.js" type="text/javascript"></script>
```

```
        <script src="../js/midi/loader.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
        <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
        <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
        <script>
          function algus(){
            MIDI.loadPlugin({
                 soundfontUrl: "./soundfont/",
                 instrument: "acoustic_grand_piano"});
          }
          let helikorgused={
            "C": 60,
                 "D": 62,
                 "E": 64,
                 "F": 65,
                 "G": 67,
                 "A": 69,
                 "H": 71
          };
          function vajutus(){
            MIDI.noteOn(0, helikorgused[document.getElementById("helistikuvalik").value], 127, 0);
            MIDI.noteOn(0, helikorgused[document.getElementById("helistikuvalik").value]+4, 127, 0.5);
            MIDI.noteOn(0, helikorgused[document.getElementById("helistikuvalik").value]+7, 127, 1);
          }
          function vajutus2(){
            MIDI.noteOn(0, helikorgused[document.getElementById("helistikuvalik").value], 127, 0);
            MIDI.noteOn(0, helikorgused[document.getElementById("helistikuvalik").value]+3, 127, 0.5);
            MIDI.noteOn(0, helikorgused[document.getElementById("helistikuvalik").value]+7, 127, 1);
          }
        </script>
</head>
<body onload="algus()">
  <select id="helistikuvalik">
     <option>C</option>
     <option>D</option>
     <option>E</option>
     <option>F</option>
     <option>G</option>
     <option>A</option>
     <option>H</option>
  </select>
  <input type="button" value="duurkolmkõla" onmousedown='vajutus()'  />
  <input type="button" value="mollkolmkõla" onmousedown='vajutus2()'  />
</body>
</html>
```

Veebiaadress

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/noot5.html`

Regiviisi mängimine

Andmed saab kätte tekstifailist XMLHttpRequesti kaudu. Tükeldatakse kõigepealt reavahetuste koha pealt iga viisi kaupa eraldi ning siis komade koha pealt nootide kaupa eraldi. Viisinoodid on kohtadel 10-26. Et enamik viise G-duuris, siis f-i MIDI-koodiks on pandud f# oma 66 - põhjalikumal juhul tasuks andmestikust järgi vaadata, et millistele nootidele vaja kõrgendused panna. Praegusel juhul mängitakse vaid esimene viis, realt number 1 - rida 0 on tulpade pealkirjade jaoks. Iga järgmine noot eelmisest poole sekundi jagu hilisem.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
       <script src="../inc/shim/Base64.js" type="text/javascript"></script>
       <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
       <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
       <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
       <script src="../js/midi/gm.js" type="text/javascript"></script>
       <script src="../js/midi/loader.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
       <script>
   let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid;

   function loeAndmed(){
     if(xhr.readyState==4){ //kui andmed kohal
      console.log(xhr.responseText);
         viisid=xhr.responseText.split("\n");
        }
   }

         function algus(){
           MIDI.loadPlugin({
               soundfontUrl: "./soundfont/",
               instrument: "acoustic_grand_piano"});
               xhr.open("GET", "regiviisid.txt", true);
           xhr.send();
         }
         let helikorgused={
           "c": 60,
               "d": 62,
               "e": 64,
               "f": 66,
               "g": 67,
               "a": 69,
               "h": 71,
               "2c": 72,
               "2d": 74,
               "2e": 76
         };
         function vajutus(){
           let rida=viisid[1].split(",");
               for(let i=0; i<16; i++){
                  MIDI.noteOn(0, helikorgused[rida[10+i]], 127, 0.5*i);
               }
               console.log(rida[10]);
         }

       </script>
</head>
<body onload="algus()">
  <input type="button" value="Esimene viis" onmousedown='vajutus()'  />

</body>
</html>
```

http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/viis1.html

Üleskirjutuste juures mitmel pool on jäänud valik mitme noodi vahel - et kuidas laulja seda parasjagu välja andis või kuivõrd me tempereeritud noodijoonestik oma ja küla kõrvade järgi kasvanud laulikuga kokku sobib. Et sageli on teine võimalik noot kantsulgudes järel, arvuti aga sellist valikuga nooti ei mõista kergesti mängida, siis jätame praegu mängides alles vaid esimese valiku.

```
        function helikorgus(rida, koht){
            let s=rida[koht];
               if(s.indexOf("[")>=0){
                  s=s.substring(0, s.indexOf("["));
                     console.log(s);
               }
               return helikorgused[s];
        }
```

Kui noodi üleskirjutuses on kandiline sulg, siis võetakse vaid sellele sulule eelnev tekst. Lõpuks leitakse helikõrguste hulgast tähele sobiv väärtus. Mitmenda viisiga tegemist, valitakse rippmenüüst.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
       <script src="../inc/shim/Base64.js" type="text/javascript"></script>
       <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
       <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
       <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
       <script src="../js/midi/gm.js" type="text/javascript"></script>
       <script src="../js/midi/loader.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
       <script>
   let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid;

   function loeAndmed(){
     if(xhr.readyState==4){ //kui andmed kohal
      console.log(xhr.responseText);
         viisid=xhr.responseText.split("\n");
        }
   }

         function algus(){
           MIDI.loadPlugin({
               soundfontUrl: "./soundfont/",
               instrument: "acoustic_grand_piano"});
               xhr.open("GET", "regiviisid.txt", true);
           xhr.send();
         }
```

```
          let helikorgused={
            "c": 60,
                "d": 62,
                "e": 64,
                "f": 66,
                "g": 67,
                "a": 69,
                "h": 71,
                "2c": 72,
                "2d": 74,
                "2e": 76
          };
          function helikorgus(rida, koht){
              let s=rida[koht];
                 if(s.indexOf("[")>=0){
                    s=s.substring(0, s.indexOf("["));
                       console.log(s);
                 }
                 return helikorgused[s];
          }
          function vajutus(){
            let rida=viisid[document.getElementById("viisivalik").value].split(",");
               let viisialgus=10;
               for(let i=0; i<16; i++){
              MIDI.noteOn(0, helikorgus(rida, viisialgus+i), 127, 0.5*i);
               }
               console.log(rida[10]);
          }

        </script>
</head>
<body onload="algus()">
  <select id="viisivalik">
     <option>1</option>
     <option>2</option>
     <option>3</option>
     <option>4</option>
     <option>5</option>
     <option>6</option>
     <option>7</option>
     <option>8</option>
     <option>9</option>
     <option>10</option>
  </select>
  <input type="button" value="Mängi viis" onmousedown='vajutus()'  />

</body>
</html>
```

Veebiviide:

http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/viis2.html

Viisi andmepuu koostamine

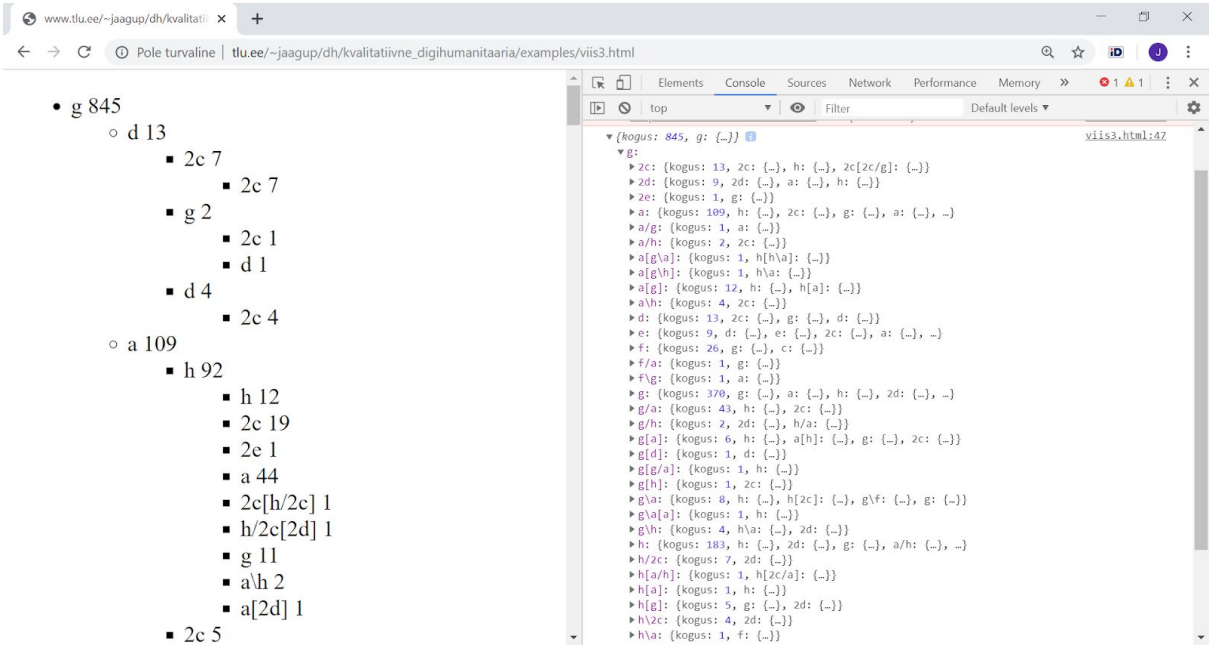
Sarnaselt eespool Pythoni abil kokku pandud puule, kannatab selle ka JavaScripti abil veebilehel teha. Siin näites võetakse aluseks g-duuris ning g-noodiga algavad viisid, igast

viisist neli algusnooti. Globaalmuutuja `andmed` on kättesaadav kogu lehe ulatuses. Puu ehitamiseks on abimuutujaks `plokk`, mis viitab parasjagu sellele kohale, kus viisipuu ehitamisega ollakse. Sarnaselt Pythoni näitele peetakse ka siin alammuutujas kogus meeles, et mitu viisi puus sellesse kohta jõuavad. Pärastine käsklus `kuvaPuu(plokk)` loob HTML-i vaste puu etteantud kohale/plokile. Kusjuures alamharude HTML-kujule panekuks kasutab sama funktsiooni väljakutset ehk rekursiooni.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
       <script src="../inc/shim/Base64.js" type="text/javascript"></script>
       <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
       <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
       <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
       <script src="../js/midi/gm.js" type="text/javascript"></script>
       <script src="../js/midi/loader.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
       <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
       <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
       <script>
   let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid, andmed;

   function loeAndmed(){
     if(xhr.readyState==4){ //kui andmed kohal
        viisid=xhr.responseText.split("\n");
         looPuu();
         document.getElementById("vastus").innerHTML=kuvaPuu(andmed);
     }
   }

     function looPuu(){
           let algusnoote=4;
              andmed={"kogus":0};
              for(let viisinr=1; viisinr<viisid.length; viisinr++){
                 rida=viisid[viisinr].split(",");
                 if(rida[6]=="g" && rida[10]=="g"){
                    plokk=andmed;
                        plokk["kogus"]+=1;
                        for(let tulp=10; tulp<10+algusnoote; tulp++){
                            if(plokk[rida[tulp]]){
                                  plokk[rida[tulp]]["kogus"]+=1;
                                } else {
                                  plokk[rida[tulp]]={"kogus": 1};
                                }
                                plokk=plokk[rida[tulp]];
                        }
                 }
              }
              console.log(andmed);
       }

       function kuvaPuu(plokk){
          let HTMLpuu="<ul>";
              for(voti in plokk){
```

```
                if(voti!="kogus"){
                 HTMLpuu+="<li>"+voti+"
"+plokk[voti]["kogus"]+kuvaPuu(plokk[voti])+"</li>";
                }
              }
           HTMLpuu+="</ul>";
           return HTMLpuu;
        }

         function algus(){
           xhr.open("GET", "regiviisid.txt", true);
           xhr.send();
         }

      </script>
</head>
<body onload="algus()">
  <div id="vastus">

  </div>
</body>
</html>
```

Väljund:
Vasakul näha andmepuu veebikujul, paremal `console.log` abil mälutõmmisena.



ja veebiaadress:
`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/viis3.html`

Mõned täiendused juurde. Puu levinumaid harusid võib olla kasulik varem näha - selleks järjestame iga haru alamväärtused nende koguste järgi kahanevalt. Ploki seest kõsime alemelemendid ehk võtmed. Noodid on need, mis pole nimega "kogus". Omakorda järjestame võtmed nende alamelementide "kogus" väärtuste järgi.

```
let votmed=Object.keys(plokk).filter(function(voti){return voti!="kogus"}).
        sort(function(v1, v2){return plokk[v2].kogus-plokk[v1].kogus});
```

```
            for(let nr=0; nr<votmed.length; nr++){
              let voti=votmed[nr];
              if(plokk[voti]["kogus"]>10){
                HTMLpuu+="<li><span onmousedown='vajutus(this)'>"+voti+"</span> "+
                    plokk[voti]["kogus"]+kuvaPuu(plokk[voti])+"</li>";
              }
```

Viiside juures on ilus ja hariv nende kõla kuulata. Et nootide nimed juba puus, siis tuleb need sealt välja korjata ja maha mängida. Pärast kuvaPuu funktsiooni tööd näeb HTML-i puu-osa välja järgnev:

```
<div id="vastus">
  <ul><li><span onmousedown="vajutus(this)">g</span> 845
     <ul><li><span onmousedown="vajutus(this)">g</span> 370
           <ul><li><span onmousedown="vajutus(this)">g</span> 149
                 <ul>
                      <li><span onmousedown="vajutus(this)">g/a</span> 75
                       <ul></ul>
                      </li>
                      <li><span onmousedown="vajutus(this)">a</span> 23<ul></ul></li>
                      <li><span onmousedown="vajutus(this)">h</span> 19<ul></ul></li>
                      ...
```

Ehk siis iga noot on omaette `<ul>`, alamnoodid `li`-de sees ning noodi nimi omaette `span`-i sees. Nii on võimalik sellele `span`-ile vajutamisele reageerida. Eelnevate nootide kätte saamiseks tuleb aga puus liikuda ülespoole. Vajutusfunktsiooni juures this on see span, millele vajutati. Sõlmest taseme ülespoole saab küsida funktsiooniga parentNode. Kontrollimaks, et kas on võimalik veel puus ülespoole liikuda, kontrollime, et kas kolm taset kõrgemal on `li`-nimeline element (vahepeal asub `ul`). Nootide nimed paigutatakse vastavanimelisse massiivi ning enne maha mängimist pööratakse massiiv ümber - sest noote asuti lisama lõpust.

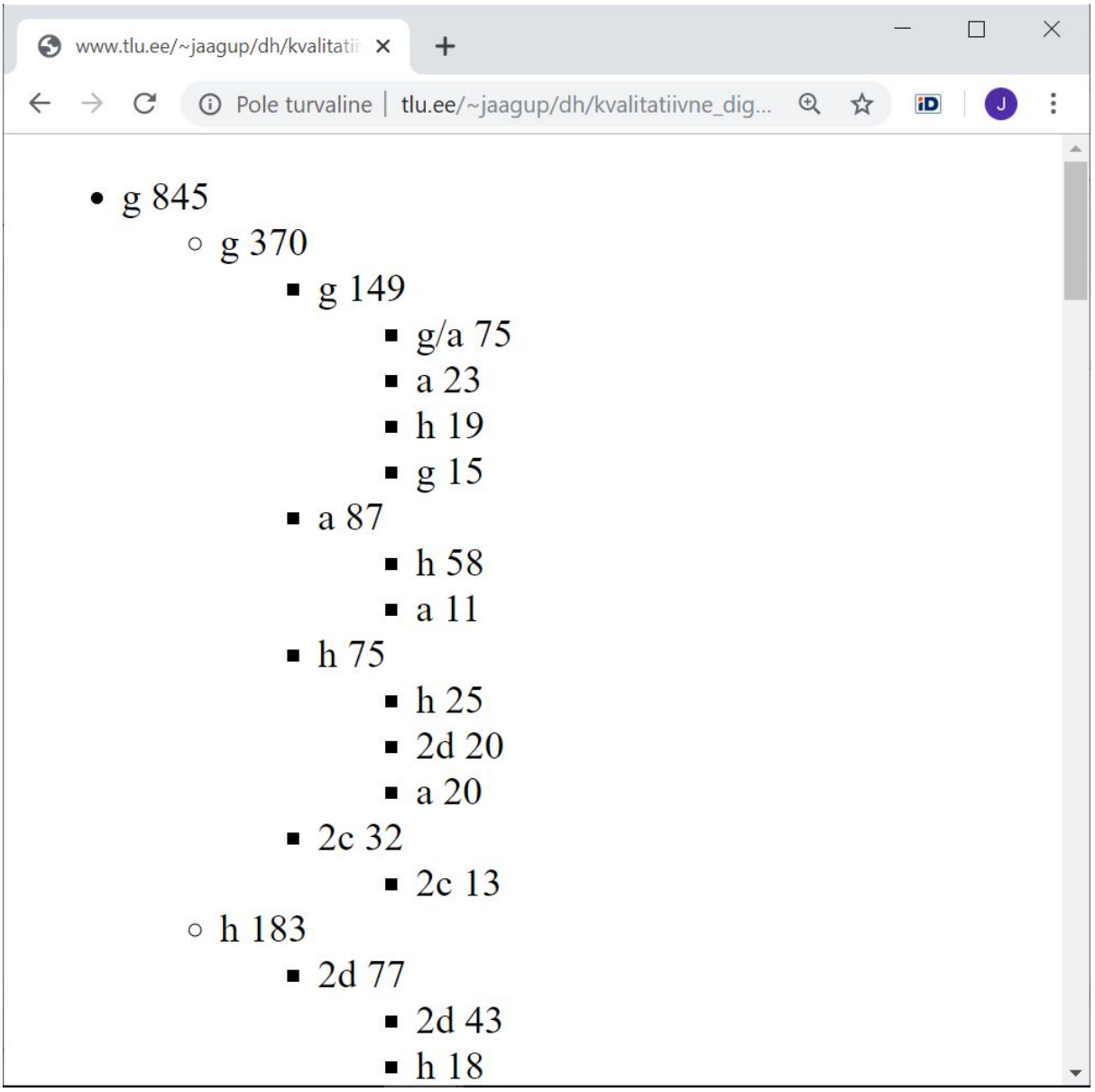
```
        function vajutus(solm){
          var noodid=[solm.innerText];
          while(solm.parentNode.parentNode.parentNode.nodeName=="LI"){
             solm=solm.parentNode.parentNode.parentNode.getElementsByTagName("span")[0];
              noodid.push(solm.innerText);
          }
          noodid=noodid.reverse();
          for(let i=0; i<noodid.length; i++){
             MIDI.noteOn(0, helikorgused[noodid[i]], 127, i*0.5)
          }
        }
```

Igale noodile pool sekundit kõlamiseks ning ongi viis kõrva juures olemas.

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
       <script src="../inc/shim/Base64.js" type="text/javascript"></script>
       <script src="../inc/shim/Base64binary.js" type="text/javascript"></script>
       <script src="../inc/shim/WebAudioAPI.js" type="text/javascript"></script>
       <script src="../js/midi/audioDetect.js" type="text/javascript"></script>
       <script src="../js/midi/gm.js" type="text/javascript"></script>
       <script src="../js/midi/loader.js" type="text/javascript"></script>
```

```
    <script src="../js/midi/plugin.audiotag.js" type="text/javascript"></script>
    <script src="../js/midi/plugin.webaudio.js" type="text/javascript"></script>
    <script src="../js/midi/plugin.webmidi.js" type="text/javascript"></script>
    <script src="../js/util/dom_request_xhr.js" type="text/javascript"></script>
    <script src="../js/util/dom_request_script.js" type="text/javascript"></script>
    <script>
let xhr=new XMLHttpRequest();
xhr.onreadystatechange=loeAndmed;
let viisid, andmed;

function loeAndmed(){
  if(xhr.readyState==4){
   console.log(xhr.responseText);
      viisid=xhr.responseText.split("\n");
   looPuu();
   document.getElementById("vastus").innerHTML=kuvaPuu(andmed);
     }
}

  function looPuu(){
        let algusnoote=4;
           andmed={"kogus":0};
           for(let viisinr=1; viisinr<viisid.length; viisinr++){
              rida=viisid[viisinr].split(",");
              if(rida[6]=="g"){
                 plokk=andmed;
                     plokk["kogus"]+=1;
                     for(let tulp=10; tulp<10+algusnoote; tulp++){
                         if(plokk[rida[tulp]]){
                               plokk[rida[tulp]]["kogus"]+=1;
                             } else {
                               plokk[rida[tulp]]={"kogus": 1};
                             }
                             plokk=plokk[rida[tulp]];
                     }
              }
           }
     }

     function kuvaPuu(plokk){
        let HTMLpuu="<ul>";
           let votmed=Object.keys(plokk).filter(function(voti){return voti!="kogus"}).
             sort(function(v1, v2){return plokk[v2].kogus-plokk[v1].kogus});
           for(let nr=0; nr<votmed.length; nr++){
             let voti=votmed[nr];
             if(plokk[voti]["kogus"]>10){
               HTMLpuu+="<li><span onmousedown='vajutus(this)'>"+voti+"</span> "+
                  plokk[voti]["kogus"]+kuvaPuu(plokk[voti])+"</li>";
             }
           }
     HTMLpuu+="</ul>";
           return HTMLpuu;
     }

     function vajutus(solm){
       var noodid=[solm.innerText];
       while(solm.parentNode.parentNode.parentNode.nodeName=="LI"){
          solm=solm.parentNode.parentNode.parentNode.getElementsByTagName("span")[0];
             noodid.push(solm.innerText);
       }
       noodid=noodid.reverse();
       for(let i=0; i<noodid.length; i++){
```

```
            MIDI.noteOn(0, helikorgused[noodid[i]], 127, i*0.5)
          }
        }

         function algus(){
           MIDI.loadPlugin({
               soundfontUrl: "./soundfont/",
               instrument: "acoustic_grand_piano"});
               xhr.open("GET", "regiviisid.txt", true);
           xhr.send();
         }
         let helikorgused={
           "c": 60,
               "d": 62,
               "e": 64,
               "f": 66,
               "g": 67,
               "a": 69,
               "h": 71,
               "2c": 72,
               "2d": 74,
               "2e": 76
         };

       </script>
</head>
<body onload="algus()">
  <div id="vastus">
  </div>
</body>
</html>
```



Veebiaadress:
`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/viis6.html`

Jooniste koostamine

Mõni tulemus ja seletus paistab paremini tekstina, mõni tabelina mõni pildina. Veebilehel jooniste kuvamine on hea moodus andmete ja mõtete illustreerimiseks. Lähemate aastate jooksul pidavat veebilehe lõuendile (`canvas`) loodav graafika üha levinumaks muutuma.

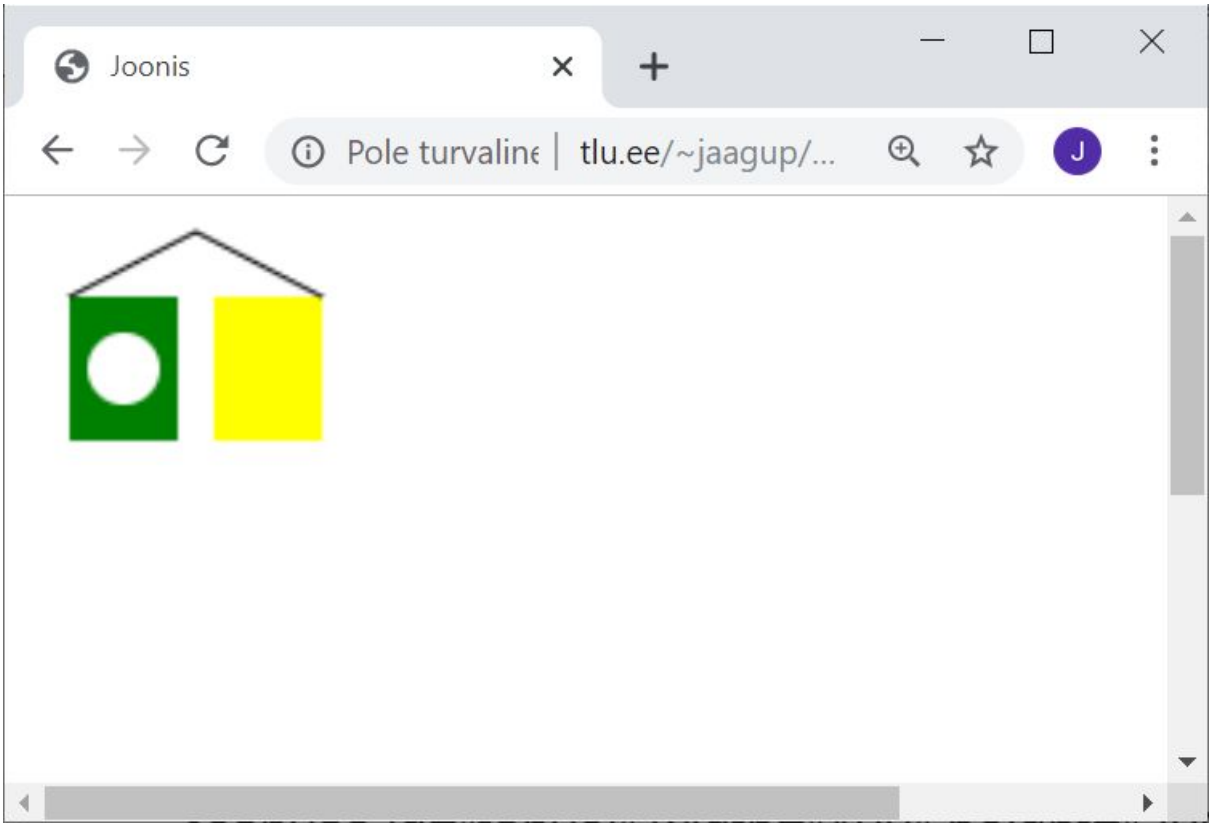
Algusnäide

Joonis pannakse tekkima kohe veebilehe avanemisel.

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <script>
          function algus(){
                 g=document.getElementById("tahvel").getContext("2d");
                  g.fillStyle="green";
                  g.fillRect(10, 20, 30, 40);
                  g.fillStyle="yellow";
                  g.fillRect(50, 20, 30, 40);
                  g.beginPath();
                  g.moveTo(10, 20);
                  g.lineTo(45, 2);
                  g.lineTo(80, 20);
                  g.stroke();
                  g.beginPath();
                  g.fillStyle="white";
                  g.arc(25, 40, 10, 0, 6.28);
                  g.fill()
          }
       </script>
  </head>
  <body onload="algus()">
    <canvas id="tahvel" width="400" height="300" /></canvas>
  </body>
</html>
```

Kuvamiseks kasutatakse graafilist konteksti, mil sees parasjagu kasutatav värv, kirja suurus ja muud vajalikud näitajad. `x`-telg on ekraanil vasakult paremale, `y`-telg ülalt alla. Käsklus `fillRect` loob ristküliku. Parameetriteks kaugus vasakult servast, kaugus ülevalt servast, ristküliku laius ja kõrgus. Soovitav värv tuleb graafilise konteksti külge panna enne kujundi joonistamist. Joone ja ringi loomine tuleb alustada `beginPath` -käsuga, lõplik vormistus vastavalt `stroke` või `fill` vastavalt sellele, kas tõmmatakse piirjooned või täidetakse seest.

Tulemus:



ja veebileht:

http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/joonis1.html

Harjutus

- Pane näide tööle
- Vaheta kujundite värve ja asukohti
- Joonista joonest ja ringist koosnev noot

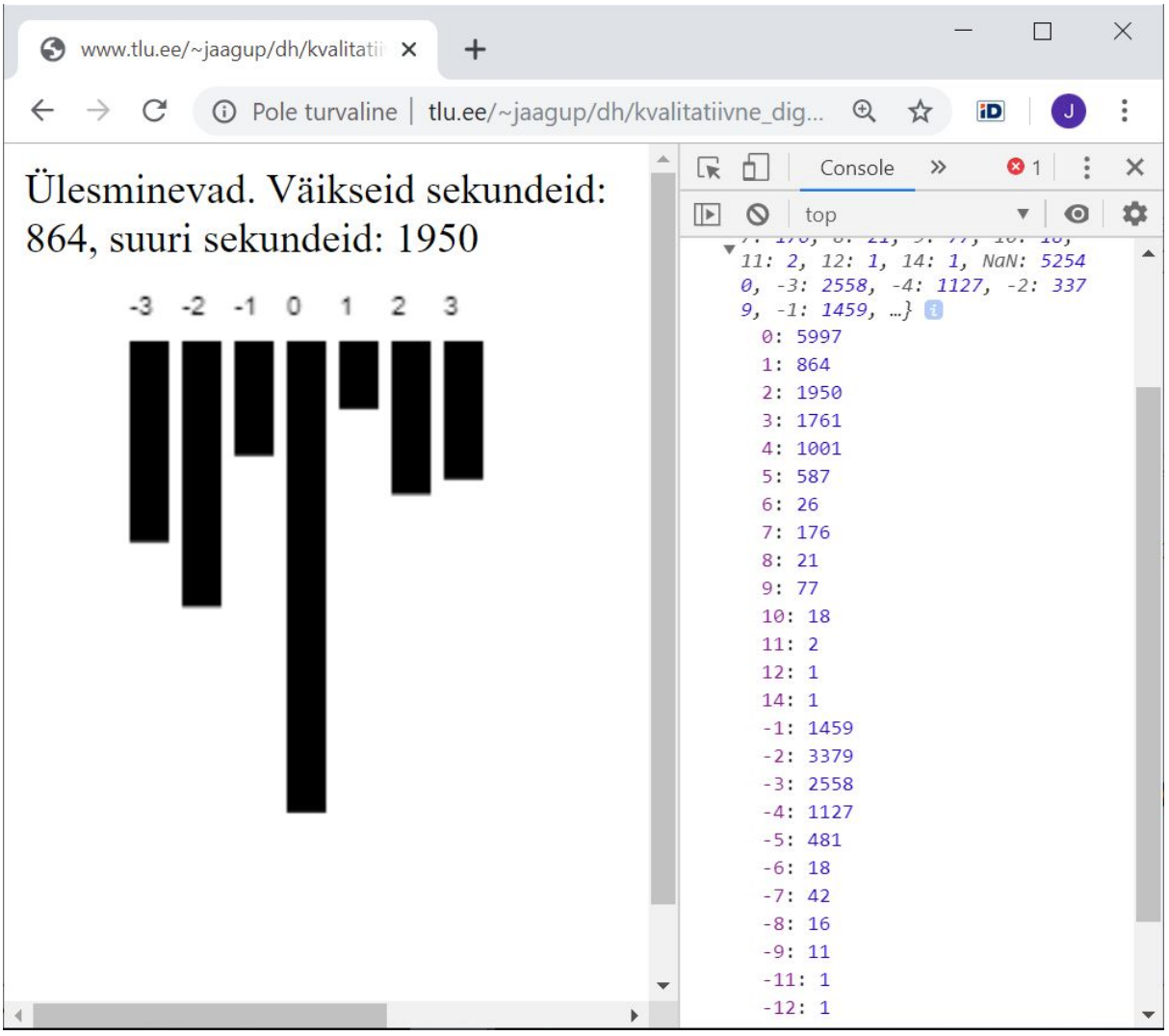
Regiviiside intervallide sagedused tulpdiagrammina

Kõigepealt leitakse nootide helikõrgused, seejärel kõrvutiste nootide kõrguste vahed ehk intervallid ning siis loetakse kokku, mitu iga helikõrgust on. Edasi kuvatakse väiksemad ja ühtlasi levinumad intervallid joonisele, kõiki andmeid näeb konsooliaknast

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <script>
   let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid, andmed;
```

```
   function loeAndmed(){
     if(xhr.readyState==4){
         viisid=xhr.responseText.split("\n");
      arvutaIntervallid();
      document.getElementById("vastus").innerHTML="Ülesminevad. Väikseid sekundeid: "+
             intervallid[1]+ ", suuri sekundeid: "+intervallid[2];
         joonistaIntervallid();
       }
   }
   function joonistaIntervallid(){
      let g=document.getElementById("tahvel").getContext("2d");
         for(intervall=-3; intervall<=3; intervall++){
            g.fillRect(100 + 20*intervall, 30, 15, intervallid[intervall]*0.03);
               g.fillText(intervall+"", 100 + 20*intervall, 20);
         }
   }
   let intervallid={};
   function arvutaIntervallid(){
       for(let viisinr=1; viisinr<viisid.length; viisinr++){
                 rida=viisid[viisinr].split(",");
                 for(let koht=0; koht<15; koht++){
                    korgus1=helikorgus(rida, koht);
                    korgus2=helikorgus(rida, koht+1);
                        intervall=korgus2-korgus1;
                        if(intervallid[intervall]){intervallid[intervall]++;}
                        else{intervallid[intervall]=1;}
                 }
        }
        console.log(intervallid);
   }
         function algus(){
              xhr.open("GET", "regiviisid.txt", true);
          xhr.send();
         }
         let helikorgused={
           "c": 60,
              "d": 62,
              "e": 64,
              "f": 66,
              "g": 67,
              "a": 69,
              "h": 71,
              "2c": 72,
              "2d": 74,
              "2e": 76
         };
         function helikorgus(rida, koht){
             let s=rida[koht];
                if(s.indexOf("[")>=0){
                   s=s.substring(0, s.indexOf("["));
                }
                return helikorgused[s];
         }
       </script>
</head>
<body onload="algus()">

  <div id="vastus">

  </div>
  <canvas id="tahvel" width="400" height="300" ></canvas>
</body>
```

```
</html>
```



http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/intervallid1.html

Harjutus

- Pane näide tööle
- Kujuta iga intervalli sagedus vastava suurusega noodina

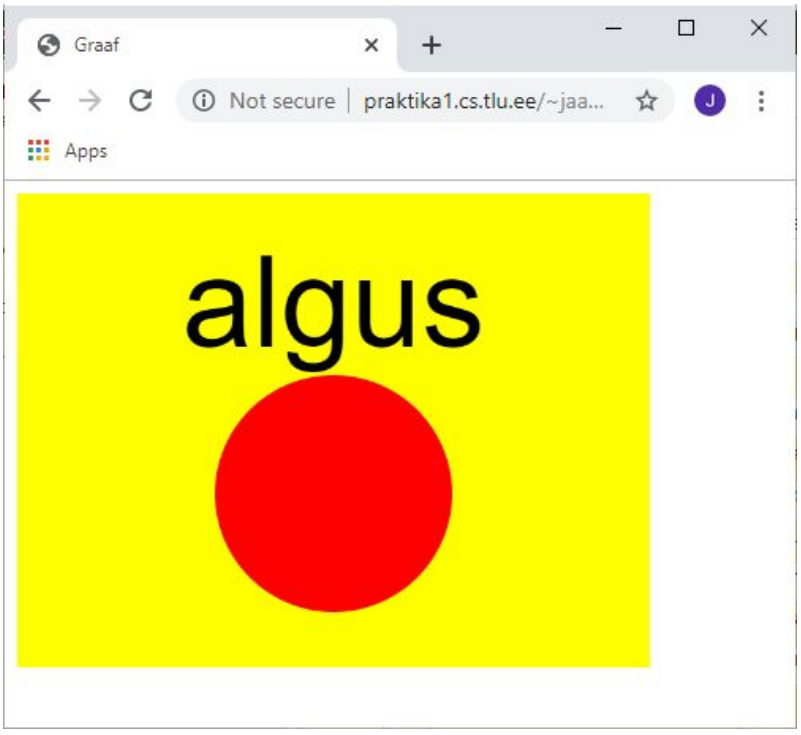
Graafijoonis Cytoscape abil

Graafiandmete veebis kuvamiseks on üheks kättesaadavaks ja mugavaks vahendiks Cytoscape, veebiaadress `js.cytoscape.org`.

Teegi võib sisse lugeda CDN-i kaudu ning kohe kasutama hakata. Graafi kuvamiseks veebilehel tuleb talle ette anda kasutatav kiht (div), mille sees siis cytoscapel lubatud vabalt toimetada. Elementidest piirdume praegu ühega, mille id on algus. Stiiliga määratakse,

millisena seda kuvatakse. Omadus `'label':'data(id)'` tähendab, et nähtavaks tekstiks on `data`-elemendi alamelement `id` väärtus.

```
<!doctype html>
<html>
  <head>
     <title>Graaf</title>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/cytoscape/3.9.4/cytoscape.min.js"></script>
        <script>
           let graaf;
           function algus(){
                graaf=cytoscape({
                   container: document.getElementById("kiht1"),
                      elements: {
                        nodes: [{data: {id: 'algus'}}]
                      },
                      style: [{selector: 'node',
                        style:{'label': 'data(id)', 'background-color': 'red'}}
                       ]
                });
              }
        </script>
  </head>
  <body onload="algus()">
    <div id="kiht1" style="width: 400px; height: 300px; background-color: yellow"></div>
  </body>
</html>
```



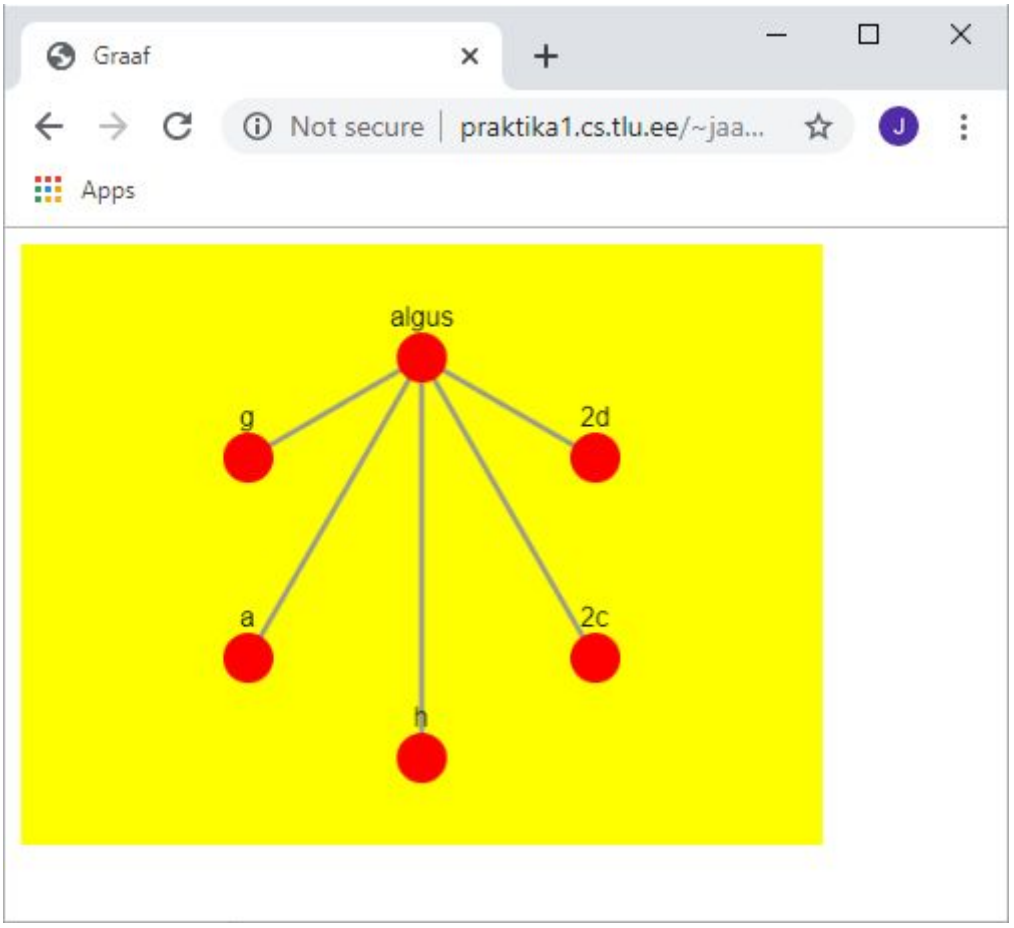
http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/graaf1.html

Rohkemate elementide näitamiseks võib need lihtsalt ritta laduda. Seoste (`edge`) puhul tuleb määrata, et kust algavad ja kuhu lõppevad. Lõpus paigutuskäsklus

```
graaf.layout({name: 'circle'}).run()
```

määrab, et sõlmed tuleb paigutada ringjoonele.

```
<!doctype html>
<html>
  <head>
     <title>Graaf</title>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/cytoscape/3.9.4/cytoscape.min.js"></script>
        <script>
           let graaf;
           function algus(){
                graaf=cytoscape({
                   container: document.getElementById("kiht1"),
                      elements: {
                        nodes: [
                          {data: {id: 'algus'}},
                          {data: {id: '2d'}},
                          {data: {id: '2c'}},
                          {data: {id: 'h'}},
                          {data: {id: 'a'}},
                          {data: {id: 'g'}}
                        ],
                        edges: [
                          {data: {source:'algus', target:'2d'}},
                          {data: {source:'algus', target:'2c'}},
                          {data: {source:'algus', target:'h'}},
                          {data: {source:'algus', target:'a'}},
                          {data: {source:'algus', target:'g'}}
                        ]
                      },
                      style: [{selector: 'node', style:{'label': 'data(id)',
                              'background-color': 'red'}}]
                });
                graaf.layout({name: 'circle'}).run()
             }
        </script>
  </head>
  <body onload="algus()">
    <div id="kiht1" style="width: 400px; height: 300px; background-color: yellow"></div>
  </body>
</html>
```



http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/graaf2.html

Graafi andmete muutmine

Ülalpool ehitasime laulu andmestiku eraldi valmis ning alles siis püüdsime seda kuvada. Siin aga on võimalik graafi ehitada koos andmetega ja kohe ka tulemust näha. Esialgu tõmmatakse andmed kohale nagu mõnes varasemas näites. Funktsiooni lisaAlgusnoot(noot) ülesandeks on lisada graafi noot kui seda veel pole ning alustuseks määrata alguse ja vastava noodi seose tugevuseks 1. Järgmisel korral suurendatakse seose omadust width ühe võrra. Noodi lisamine

```
graaf.add({group:'nodes', data:{id:noot}});
```

Seose lisamine

```
graaf.add({group:'edges', data:{source:'algus', target:noot, width:1}});
```

Seoste küsimine, mille sihtkohaks on soovitud noot

```
let e=graaf.edges("[target='"+noot+"']");
```

Omaduse width suurendamine ühe võrra. Ehk siis küsitakse vana väärtus, arvutatakse üks juurde ning paigutatakse samale kohale tagasi

```
e.data("width", e.data("width")+1);
```

Funktsioon tervikuna

```
function lisaAlgusnoot(noot){
   if(kasutusel.includes(noot)){
         if(graaf.getElementById(noot).length==0){
               graaf.add({group:'nodes', data:{id:noot}});
               graaf.add({group:'edges', data:{source:'algus', target:noot, width:1}});
            } else {
               let e=graaf.edges("[target='"+noot+"']");
                  e.data("width", e.data("width")+1);
            }
       }
}
```

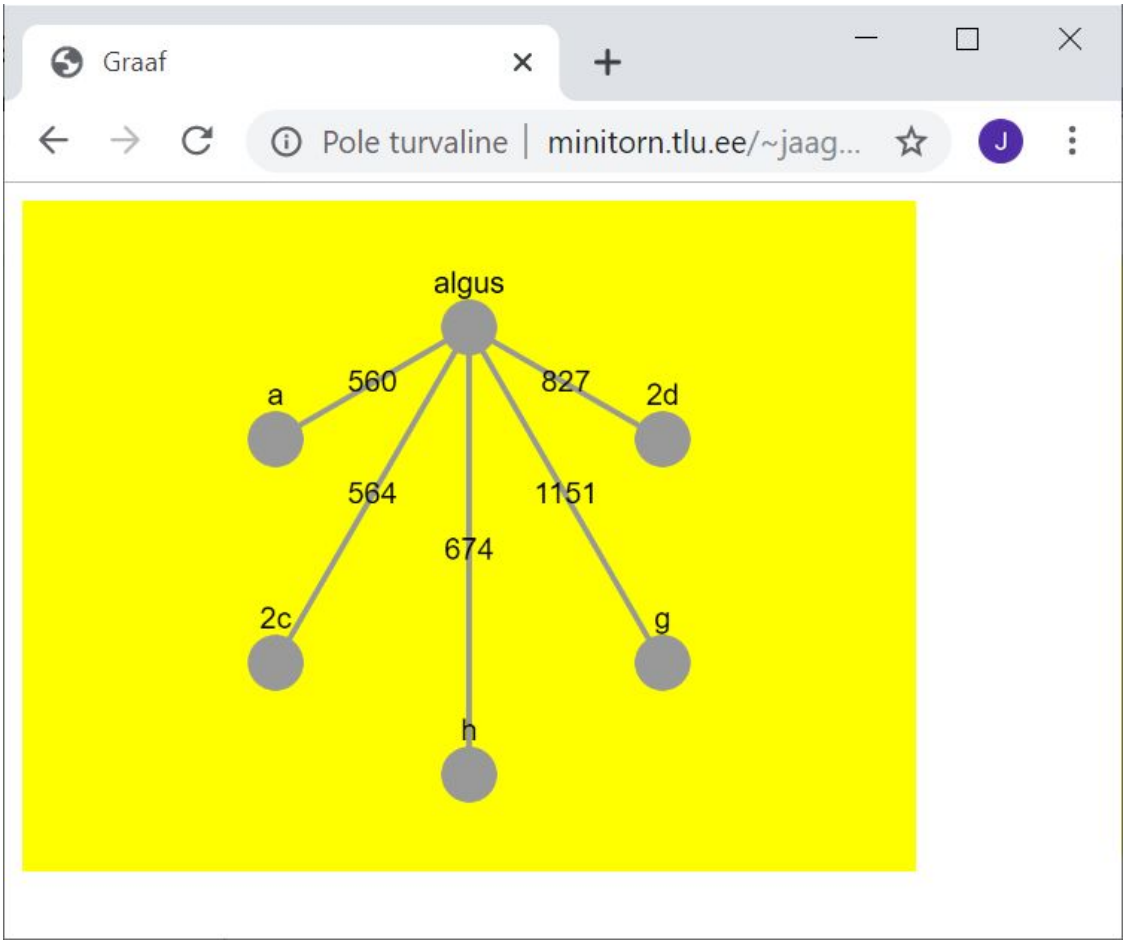
Seose peal teksti kuvamiseks tuleb lihtsalt määrata, millisest tunnusest see võetakse

```
{selector: 'edge', style:{'label': 'data(width)'}}
```

Rakenduse kood tervikuna

```
<!doctype html>
<html>
  <head>
     <title>Graaf</title>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/cytoscape/3.9.4/cytoscape.min.js"></script>
        <script>
           let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid, andmed, kasutusel=["g", "a", "h", "2c", "2d"];

   function loeAndmed(){
     if(xhr.readyState==4){
         viisid=xhr.responseText.split("\n");
         for(i=1; i<viisid.length; i++){
            lisaAlgusnoot(viisid[i].split(",")[10])
         }
                graaf.layout({name: 'circle'}).run()

        }
   }

   function lisaAlgusnoot(noot){
      if(kasutusel.includes(noot)){
            if(graaf.getElementById(noot).length==0){
                  graaf.add({group:'nodes', data:{id:noot}});
                  graaf.add({group:'edges', data:{source:'algus', target:noot, width:1}});
                     console.log(noot);
               } else {
                  let e=graaf.edges("[target='"+noot+"']");
                     e.data("width", e.data("width")+1);
               }
```

```
        }
    }
          let graaf;
          function algus(){
               graaf=cytoscape({
                  container: document.getElementById("kiht1"),
                     elements: {
                       nodes: [
                         {data: {id: 'algus'}},
                       ],
                     },
                     style: [
                      {selector: 'node', style:{'label': 'data(id)'}},
                      {selector: 'edge', style:{'label': 'data(width)'}},
                    ]
               });
             xhr.open("GET", "regiviisid.txt", true);
          xhr.send();
             }
        </script>
  </head>
  <body onload="algus()">
    <div id="kiht1" style="width: 400px; height: 300px; background-color: yellow"></div>
  </body>
</html>
```



http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/graaf4.html

Harjutus

- Pane näited tööle
- Koosta veebilehel tähtede järgnevusseoste graaf sõnadele "sina", "siia" ja "saia".

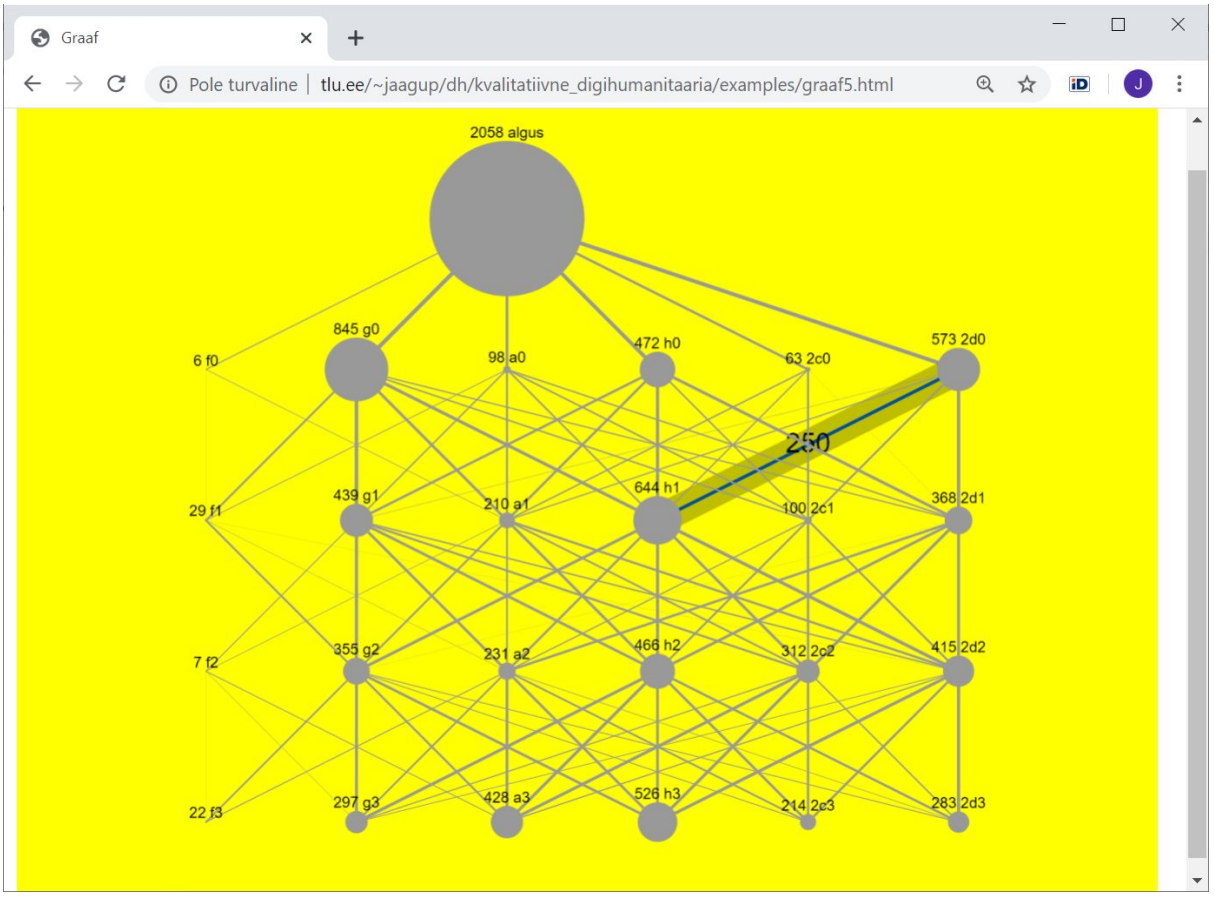
Viisi nootide järgnevus

Veidi pikem näide veebilehel töötava graafi võimaluste kohta. Koostatakse lugude soovitud arvust algusnootidest, pildi selguse huvides määratakse kasutusel olevad ehk nähtavad noodid. Noodisõlmede ning nendevaheliste seoste suurus sõltub esinemiskordade arvust. Seoste puhul kasutatakse logaritmi, et eri suurusega seosed näha jääksid. Praegu piirdutakse g-duuri viisidega. Seose peal hiirega vajutades näeb seose esinemiskordade arvu.

```
<!doctype html>
<html>
  <head>
     <title>Graaf</title>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/cytoscape/3.9.4/cytoscape.min.js"></script>
        <script>
           let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid, andmed, kasutusel=["f", "g", "a", "h", "2c", "2d"];
   let pikkus=4;
   let nalg=10;
   function loeAndmed(){
     if(xhr.readyState==4){
         viisid=xhr.responseText.split("\n");
         for(i=1; i<viisid.length; i++){
            let v=viisid[i].split(",");
               if(v[6]=="g"){
                console.log(i);
             lisaAlgusnoot(v[nalg])
                for(noodinr=0; noodinr<pikkus-1; noodinr++){
                 if(kasutusel.includes(v[nalg+noodinr]) &&
kasutusel.includes(v[nalg+noodinr+1])){
                    lisaNoodipaar(v[nalg+noodinr]+noodinr, v[nalg+noodinr+1]+(noodinr+1));

                 }
                }
               }
         }
         paiguta();
                graaf.layout({name: 'preset'}).run()
//               graaf.layout({name: 'breadthfirst'}).run()

        }
   }

   function lisaAlgusnoot(algnoot){
```

```
      if(kasutusel.includes(algnoot)){
                let algn=graaf.nodes("[id='algus']");
                algn.data("width", algn.data("width")+1);
            let noot=algnoot+"0";
            if(graaf.getElementById(noot).length==0){
                  graaf.add({group:'nodes', data:{id:noot, width:1}});
                  graaf.add({group:'edges', data:{source:'algus', target:noot,
                            width:1, logwidth:Math.log(1)+0.1
                             }});
                } else {
                  let n=graaf.nodes("[id='"+noot+"']");
                     n.data("width", n.data("width")+1);
                  let e=graaf.edges("[target='"+noot+"']");
                     e.data("width", e.data("width")+1);
                  e.data("logwidth", Math.log(e.data("width"))/2.5);
                }
          }
   }

   function lisaNoodipaar(n1, n2){
      if(graaf.getElementById(n1).length==0){
            graaf.add({group:'nodes', data:{id:n1, width:1}});
          }
      if(graaf.getElementById(n2).length==0){
            graaf.add({group:'nodes', data:{id:n2, width:1}});
          }
          let e=graaf.edges("[source='"+n1+"'][target='"+n2+"']");
          if(e.length==0){
                graaf.add({group:'edges', data:{source:n1, target:n2, width:1,
logwidth:Math.log(1)+0.1}});
          } else {
                let n=graaf.nodes("[id='"+n2+"']");
             n.data("width", n.data("width")+1);
                e.data("width", e.data("width")+1);
                e.data("logwidth", Math.log(e.data("width"))/2.5);
          }
   }

   function paiguta(){
      for(nr=0; nr<=pikkus; nr++){
            for(koht=0; koht<kasutusel.length; koht++){
                  let n=graaf.getElementById(kasutusel[koht]+nr);
                     n.position("x", koht*100);
                     n.position("y", nr*100);
                }
          }
          let a=graaf.getElementById("algus");
          a.position("x", 200);
          a.position("y", -100);
   }
           let graaf;
           function algus(){
                graaf=cytoscape({
                   container: document.getElementById("kiht1"),
                      elements: {
                        nodes: [
                          {data: {'id': 'algus', 'width': 1}},
                        ],
                      },
                      style: [
                       {selector: 'node', style:{'label': 'data(id)', 'font-size': '10px',
                           'width': function(ele){return ele.data("width")/20},
```

```
                                'height':function(ele){return ele.data("width")/20},
                                'content': function(ele){return ele.data("width")+"
"+ele.data("id");}
                                }
                                },
                        {selector: 'edge:inactive', style:{'width':'data(logwidth)'}},
                        {selector: 'edge:active', style:{'label': 'data(width)',
                                'width':'data(logwidth)',         'font-size':'18px'}},

                    ]
                });
            xhr.open("GET", "regiviisid.txt", true);
        xhr.send();
            }
        </script>
  </head>
  <body onload="algus()">
    <div id="kiht1" style="width: 700px; height:500px; background-color: yellow"></div>
  </body>
</html>
```



Kasutatav aadressil

http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/graaf5.html

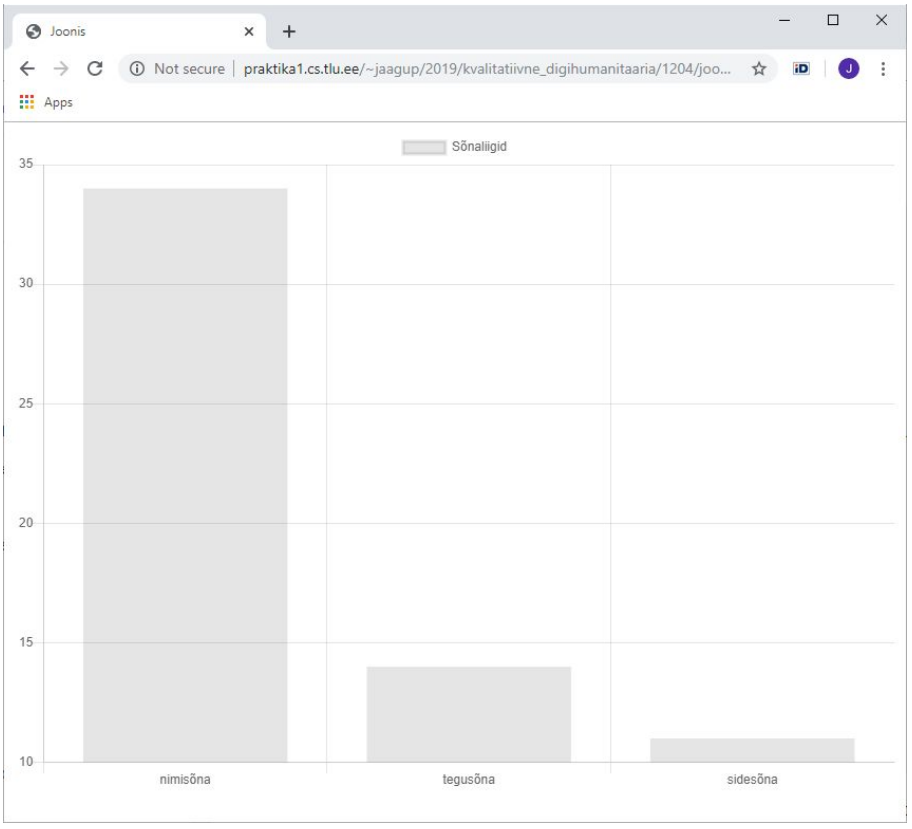
Chart.js

Joonised aitavad arve ilmestada. Püsivate andmetega joonised saab pildina teksti sisse panna. Kui algandmed täienevad või kasutaja tasub pakkuda võimalus andmetega mängida ja neid mitmest küljest vaadata, siis aitavad veebilehel genereeritud joonised. Üheks lihtsaks põhijooniseid pakkuvaks JavaScripti teegiks on Chart.js.

Joonise saamiseks tuleb teegi kood CDNi ehk koodipakkuja juurest sisse lugeda, määrata joonise arvud ja selgitavad sõnad, koostada lehel element joonise paigutamiseks ja siduda see joonisega.

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
         function algus(){
           new Chart(document.getElementById("joonisekiht"), {
                 type:'bar',
                 data:{
                    labels:['nimisõna', 'tegusõna', 'sidesõna'],
                        datasets:[{
                          label: 'Sõnaliigid',
                          data:[34, 14, 11]
                        }]
                 }
              })
         }
       </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 400px; height:300px;"></canvas>
  </body>
</html>
```

Nii võibki tulemust imetleda



Harjutus

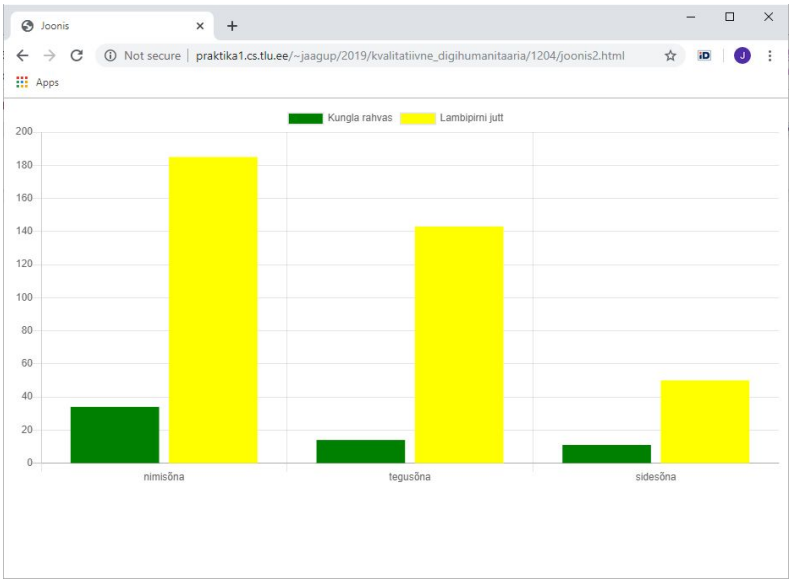
- Tee näide läbi
- Koosta tulpdiagramm sõnade ja nende tähtede arvudega

Värvide lisamine, mitu andmerida

Andmestiku all muutuja datasets on massiiv, sinna alla saab paigutada mitu andmerida, igaühega neist kaasa anda ka soovitud taustavärvi

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
         function algus(){
           new Chart(document.getElementById("joonisekiht"), {
                 type:'bar',
                 data:{
                    labels:['nimisõna', 'tegusõna', 'sidesõna'],
                        datasets:[{
                          label: 'Kungla rahvas',
                          data:[34, 14, 11],
                             backgroundColor: 'green'},
               {label: 'Lambipirni jutt',
```

```
                        data:[185, 143, 50],
                           backgroundColor: 'yellow'}
                      ]
                  }
               })
         }
      </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 85%;"></canvas>
  </body>
</html>
```

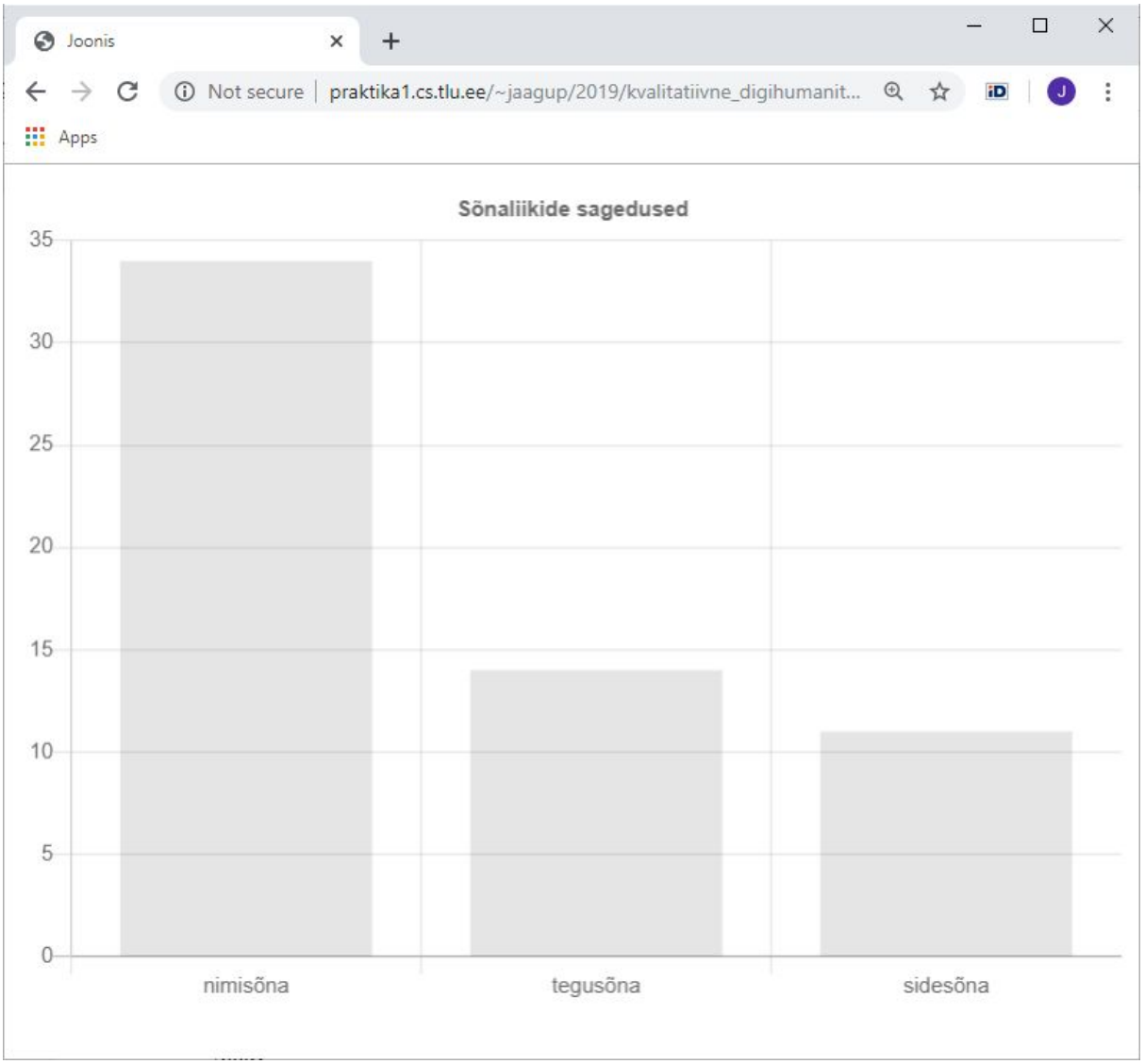


Mõned seadistused: legend peitu ning y-telje loendamine algab nullist. Nii paremini aru saada, kui üks tulp on teisest mingi arv kordi suurem või väiksem. Kogu joonisele ka pealkiri.

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
         function algus(){
           new Chart(document.getElementById("joonisekiht"), {
                 type:'bar',
                 data:{
                    labels:['nimisõna', 'tegusõna', 'sidesõna'],
                        datasets:[{
                          label: 'Sõnaliigid',
                          data:[34, 14, 11]
                        }]
                 },
                 options:{
                   legend:{
                        display: false
                      },
```

```
                    scales:{
                         yAxes:[{
                           ticks:{
                                beginAtZero: true
                              }
                           }]
                        },
                        title:{
                          display: true,
                          text: 'Sõnaliikide sagedused'
                        }
                    }
                })
            }
        </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 400px; height:300px;"></canvas>
  </body>
</html>
```

Tulemus:



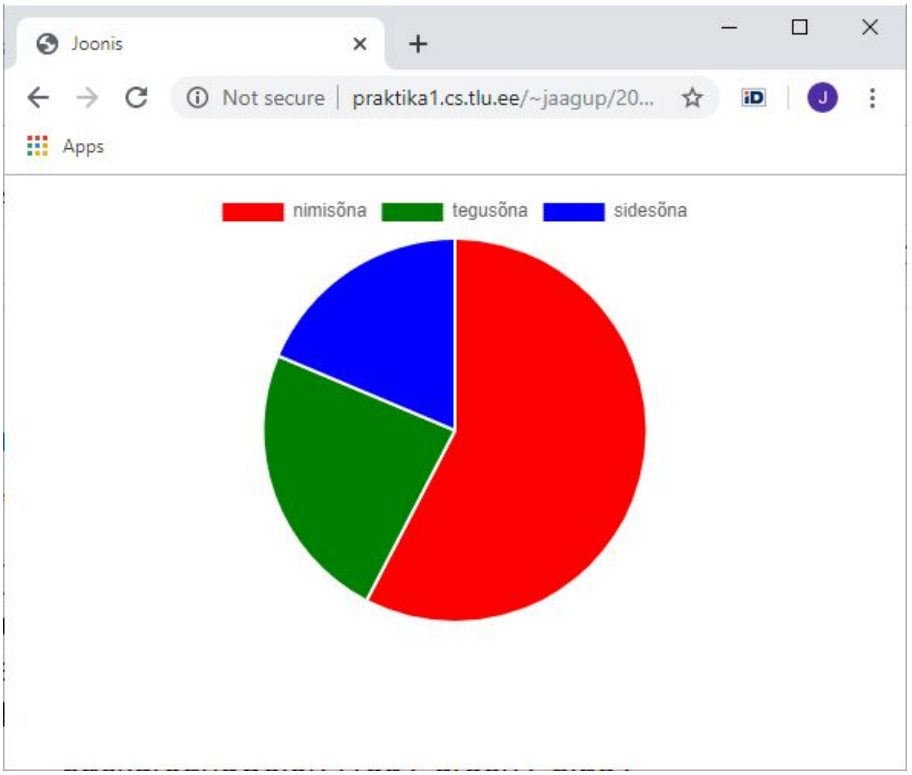
Harjutus

- Tee näide läbi
- Koosta sõnadest tulpdiagramm, kus iga sõna kohta üks tulp näitab täishäälikute arvu ja teine sulghäälikute arvu
- Pane tulpade skaala algama nullist

Sektordiagramm

Kasutatakse, kui soovitakse näidata, kui suure osa moodustab iga väärtus tervikust. Tehniliselt loomine tulpdiagrammi moodi, ainult et joonise tüübiks tuleb valida "pie".

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
         function algus(){
           new Chart(document.getElementById("joonisekiht"), {
                 type:'pie',
                 data:{
                    labels:['nimisõna', 'tegusõna', 'sidesõna'],
                        datasets:[{
                          label: 'Sõnaliigid',
                          data:[34, 14, 11],
                              backgroundColor: ['red', 'green', 'blue']
                        }]
                 }
              })
         }
       </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 85%;"></canvas>
  </body>
</html>
```

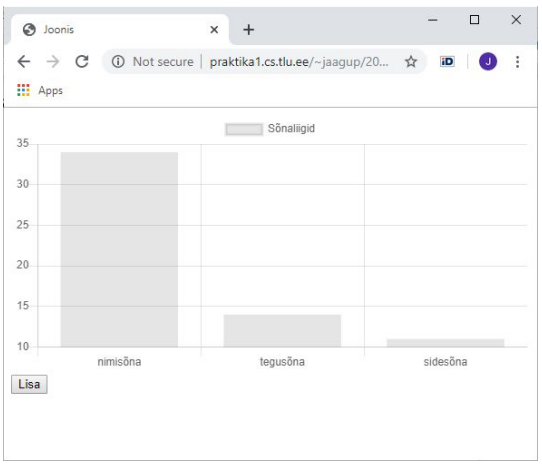


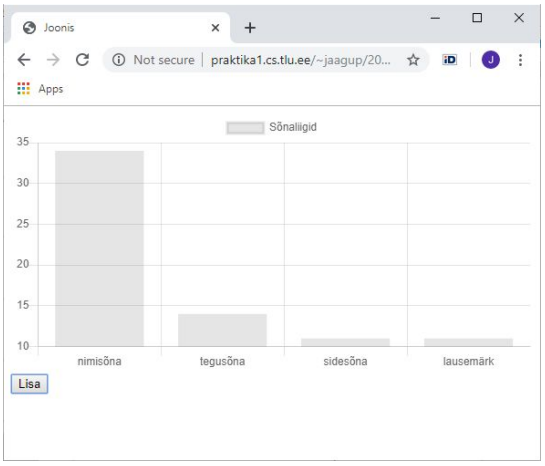
Jooniste andmestikku saab jooniste näitamise ajal muuta. All näites lisatakse tunnuseks `lausemärk` ning tema väärtuseks `11`. Käsu `update()` peale kuvatakse joonis uute andmetega.

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
         let joonis;
         function algus(){
           joonis=new Chart(document.getElementById("joonisekiht"), {
                  type:'bar',
                  data:{
                     labels:['nimisõna', 'tegusõna', 'sidesõna'],
                         datasets:[{
                           label: 'Sõnaliigid',
                           data:[34, 14, 11]
                         }]
                  }
               })
         }
         function vajutus(){
            joonis.data.labels.push("lausemärk");
                joonis.data.datasets[0].data.push(11);
                joonis.update();
         }
       </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 85%;"></canvas>
```

```
        <input type="button" value="Lisa" onclick="vajutus()" />
  </body>
</html>
```

Pildid seisust enne ja pärast:



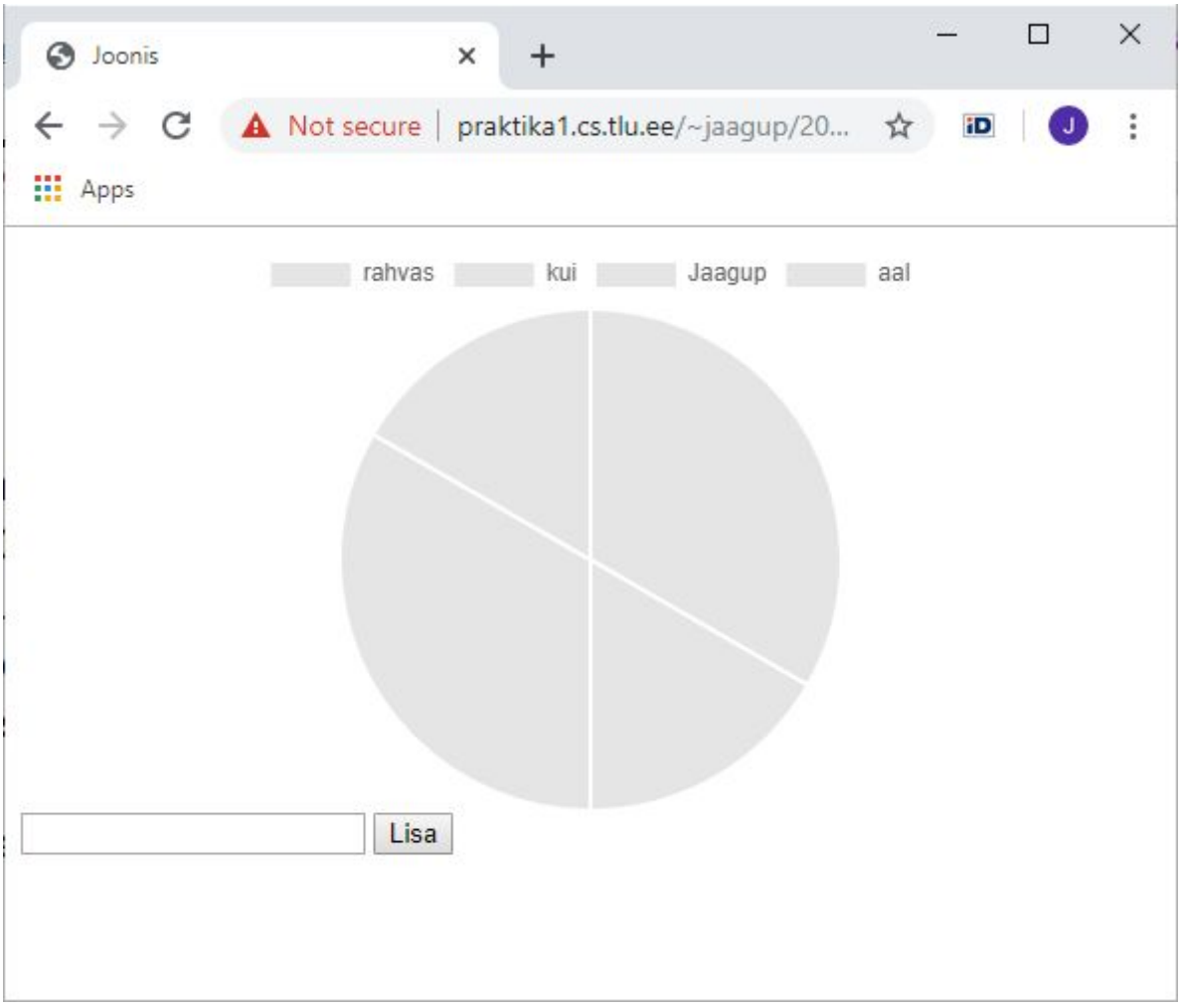


Harjutus

- Koostage sektordiagramm sõnadega "rahvas" ja "kui", sektori suurus vastab sõna tähtede arvule.
- Kasutaja saab tekstiväljast ükshaaval lisada sõnu, sõna jõuab sektordiagrammile, sektori suurus vastab sõna tähtede arvule.

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
         let joonis;
         function algus(){
           joonis=new Chart(document.getElementById("joonisekiht"), {
                 type:'pie',
                 data:{
                    labels:['rahvas', 'kui'],
                       datasets:[{
                         label: 'Sõnade tähtede arv',
                         data:[6, 3]
                       }]
                 }
              })
         }
         function vajutus(){
            let s=document.getElementById("tekst1").value;
               if(s.trim().length==0){return;}
            if(joonis.data.labels.includes(s)){return;}
               joonis.data.labels.push(s);
```

```
                joonis.data.datasets[0].data.push(s.length);
                joonis.update();
                document.getElementById("tekst1").value="";
            }
        </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 85%;"></canvas>
        <input type="text" id="tekst1" />
        <input type="button" value="Lisa" onclick="vajutus()" />
  </body>
</html>
```



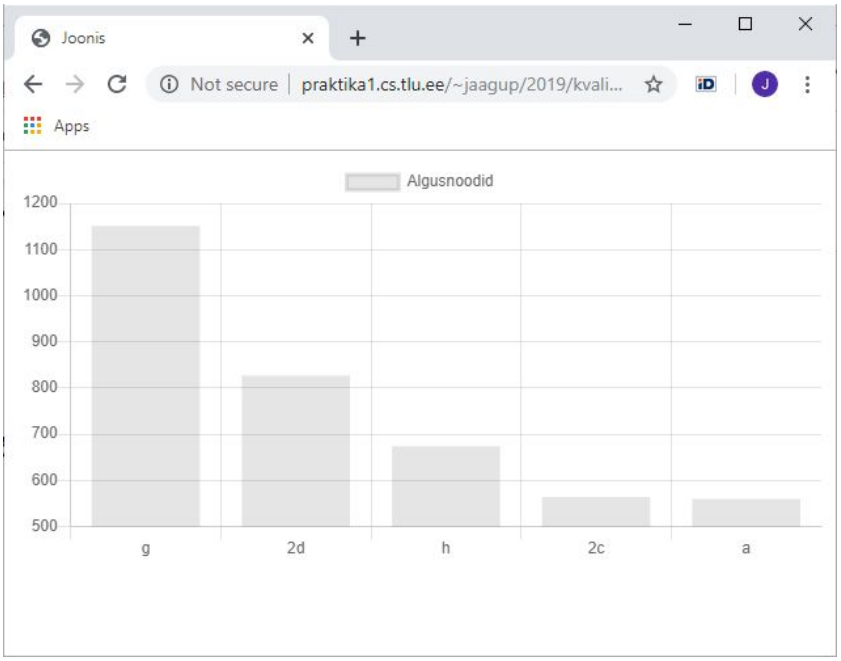
Joonis veebist loetud andmete põhjal

XmlHttpRequest-i kaudu loetud andmetest saab koha peal arvutada sobivad väärtused ning need joonisena välja kuvada. Siin kuvatakse tulpdiagrammina soovitud arvu levinumate algusnootide sagedused

```
<!doctype html>
```

```
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
   let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid, joonis;
   let mituKuvada=5;

   function loeAndmed(){
     if(xhr.readyState==4){
      console.log(xhr.responseText);
         viisid=xhr.responseText.split("\n");
      arvutaAlgusnoodid();
        }
   }
   function arvutaAlgusnoodid(){
      kogused={}
         for(let i=1; i<viisid.length; i++){
            let noot=viisid[i].split(",")[10];
                if(kogused[noot]){kogused[noot]++;}
                else{kogused[noot]=1;}
         }
         jarjestatud=Object.keys(kogused).sort(
            function(v1, v2){return kogused[v2]-kogused[v1]});
         for(nr=0; nr<mituKuvada; nr++){
           voti=jarjestatud[nr];
              joonis.data.labels.push(voti);
              joonis.data.datasets[0].data.push(kogused[voti])
         }
         joonis.update();
   }
         function algus(){
           joonis=new Chart(document.getElementById("joonisekiht"), {
                 type:'bar',
                 data:{
                    labels:[],
                        datasets:[{
                          label: 'Algusnoodid',
                          data:[]
                        }]
                 }
              });
              xhr.open("GET", "regiviisid.txt", true);
           xhr.send();
         }
       </script>
  </head>
  <body onload="algus()">
     <canvas id="joonisekiht" style="width: 85%;"></canvas>
  </body>
</html>
```

Tulemus:



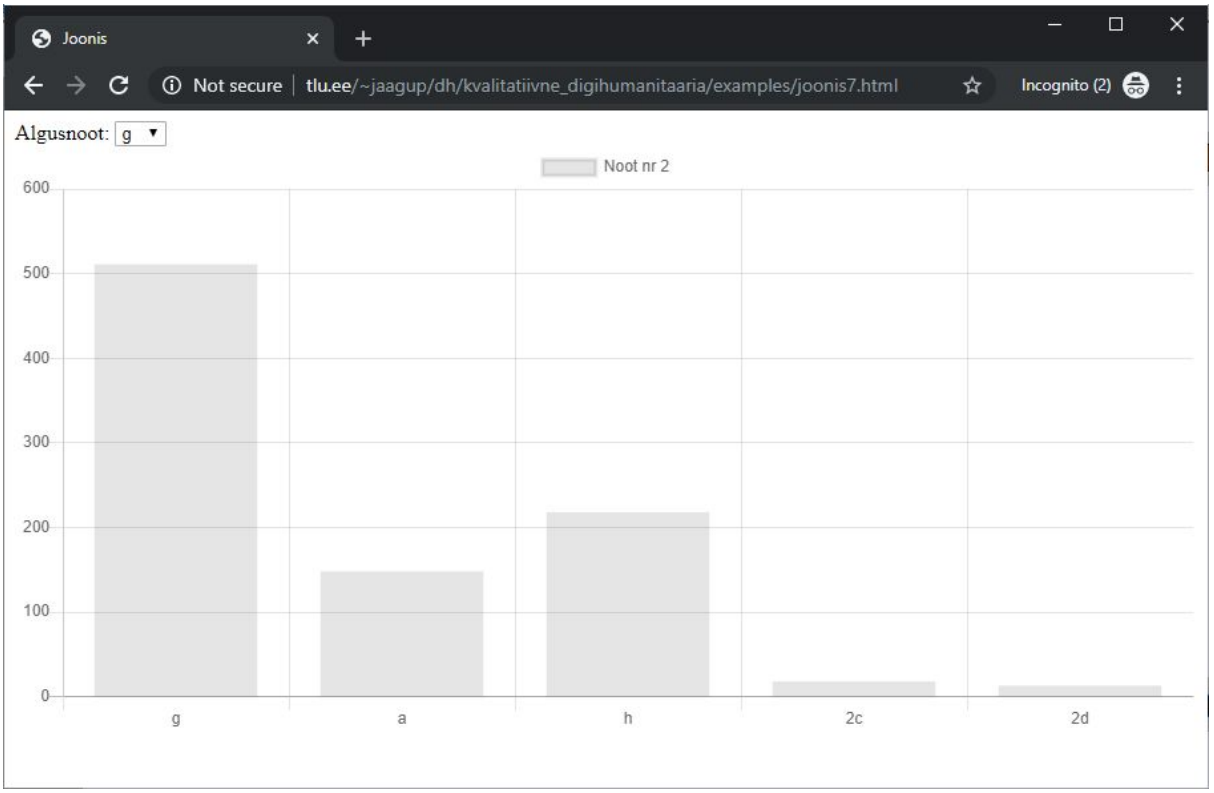
Leht nähtav aadressil

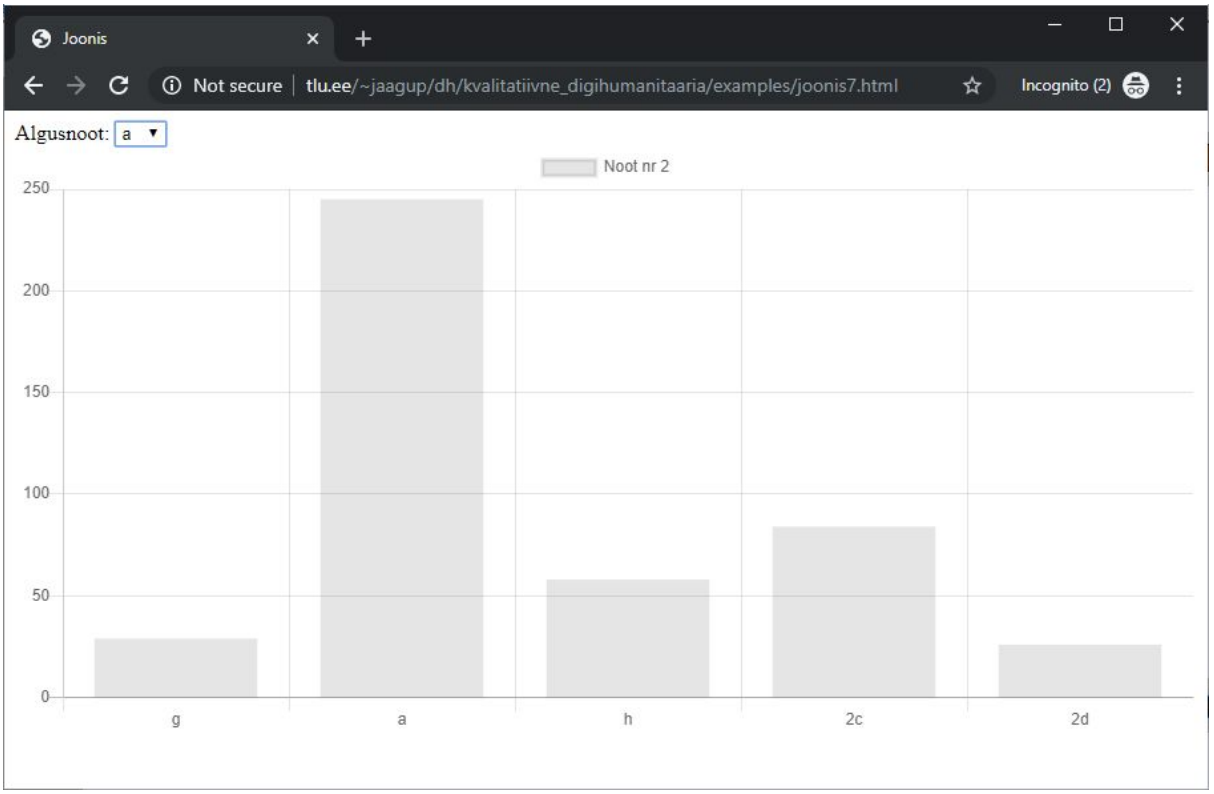
`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/joonis6.html`

Andmetega mängimiseks saab kasutaja valida algusnoodi ning tulpdiagrammil näidatakse talle loo teise noodi esinemise sagedused tulpdiagrammina.

```
<!doctype html>
<html>
  <head>
    <title>Joonis</title>
       <meta charset="utf-8" />
       <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
       <script>
   let xhr=new XMLHttpRequest();
   xhr.onreadystatechange=loeAndmed;
   let viisid, joonis;
   let mituRippmenyys=5;
   let jooniselKuvada=["g", "a", "h", "2c", "2d"];

   function loeAndmed(){
     if(xhr.readyState==4){
         viisid=xhr.responseText.split("\n");
      arvutaAlgusnoodid();
        }
   }
   function arvutaAlgusnoodid(){
      let kogused={}
         for(let i=1; i<viisid.length; i++){
            let noot=viisid[i].split(",")[10];
               if(kogused[noot]){kogused[noot]++;}
               else{kogused[noot]=1;}
         }
         let jarjestatud=Object.keys(kogused).sort(
            function(v1, v2){return kogused[v2]-kogused[v1]});
```

```
        let t="";
        for(nr=0; nr<mituRippmenyys; nr++){
          voti=jarjestatud[nr];
              t+="<option>"+voti+"</option>";
        }
        document.getElementById("valik1").innerHTML=t;
        noodidVastavaltValikule();
   }
   function noodidVastavaltValikule(){
      let valitud=document.getElementById("valik1").value;
        let kogused={}
        for(let i=1; i<viisid.length; i++){
           let m=viisid[i].split(",");
              if(m[10]==valitud){
            let noot=m[11];
               if(kogused[noot]){kogused[noot]++;}
               else{kogused[noot]=1;}
              }
        }
        joonis.data.labels.length=0;
        joonis.data.datasets[0].data.length=0;
        for(let nr=0; nr<jooniselKuvada.length; nr++){
             joonis.data.labels.push(jooniselKuvada[nr]);
             joonis.data.datasets[0].data.push(kogused[jooniselKuvada[nr]])
        }
        joonis.update();
   }
        function algus(){
          joonis=new Chart(document.getElementById("joonisekiht"), {
                type:'bar',
                data:{
                   labels:[],
                      datasets:[{
                        label: 'Noot nr 2',
                        data:[]
                      }]
                }
             });
             xhr.open("GET", "regiviisid.txt", true);
          xhr.send();
        }
      </script>
  </head>
  <body onload="algus()">
       Algusnoot: <select id="valik1" onchange="noodidVastavaltValikule()"></select><br
/>
     <canvas id="joonisekiht" style="width: 85%;"></canvas>
  </body>
</html>
```





Tulemus veebis

`http://www.tlu.ee/~jaagup/dh/kvalitatiivne_digihumanitaaria/examples/joonis7.html`

Kordamisküsimused

Märksõnu seminariks, kvalitatiivne digihumanitaaria

Andmepuude kasutamine järjestatud andmete seaduspärade leidmiseks ja esitamiseks.
Bi-, tri- ja tetragramid. (Suhteliste) sageduste võrdlemine, autori andmete tuvastamine
Puukujuliste andmete esitamise vormingud: json, XML.
Andmepuu loomise algoritm
Näiteid eri tüüpi andmetest koostatud puudest -
  tähed, silbid, sõnaliigid, lauseliikmed, viisinoodid
Teksti lauseliikmete leidmine, kasutamine puu koostamisel.
Puu harudes jälgimine, mille põhjal vastav haru moodustati
Puus elementide kuvamine sageduse järjekorras
Regiviiside andmestikust nootide eraldamine
XML-väljund, xml.dom.minidom.
 HTML-väljund <ul>,
 Smart Art-i abil andmepuu kuvamine
Iseloomulike väärtuste esile toomine hii-ruut testi abil (võtmesõnad)

Graafi kasutamine andmete esitamisel ja uurimisel, näiteid
Gephi abil graafide loomine, paigutusmoodused Gephis
Viisijärgnevuste esitamine graafina
Graafidega seotud animatsioonide loomine
Graafid veebilehel

Väliste veebiandmete kasutamine oma rakenduste juures
Teenuste näiteid
Tehnilisi kasutusnäiteid
Oma andmete veebi kaudu kättesaadavaks tegemine

Kaardirakendused, kasutatavad vahendid, võimalused
Koordinaatide leidmise moodused
Interaktiivse kaardirakenduse koostamine

MIDI helid (veebilehel)
Graaf veebilehel
Joonised veebilehel

Kokkuvõte

Kvalitatiivsed meetodid võimaldavad samu nähtusi kirjeldada mitmest suunast ning saada tulemustele kinnitusi mitmel moel samadele järeldustele jõudmise kaudu. Või ka saada vaadeldes erisuguseid tulemusi ning sealtkaudu teha järeldusi, et uuritav aines on esmapilgul paistvast keerukam ning mitmekülgsemaks kasutamiseks vajab pikemat süvenemist. Aastatega arenevad digivahendid võimaldavad nüüdseks ka humanitaaridel oma tööle kasulikke abijõude leida selleks samas liialt tehnilistesse üksikasjadesse uppumata. Siinse kirjutise näited annavad mõne suuna jaoks ette otsesed vahendid pealispinna alla piilumiseks, teisalt jälle loodetavasti julgustavad konkreetsele mure lahendamiseks sobivaid abivahendeid otsima.